JAPAN
SMARTPHONE
SECURITY
ASSOCIATION

Android Application Secure Design/Secure Coding Guidebook



February 1, 2017 Edition

Japan Smartphone Security Association (JSSEC)

Secure Coding Working Group

Document control number: JSSEC-TECA-SC-GD20170201BE

目錄

安卓应用安全指南 中文版

原文：Android Application Secure Design/Secure Coding Guidebook

译者：飞龙

版本：2017.2.1

自豪地采用谷歌翻译

- 在线阅读
- PDF格式
- EPUB格式
- MOBI格式
- Github

赞助我



协议

一、简介

（略）

二、本书结构

（略）

三、安全设计和编程的基础知识

（略）

四、以安全方式使用技术

4.1 创建或使用活动

4.1.1 示例代码

使用活动的风险和对策取决于活动的使用方式。 在本节中，我们根据活动的使用情况，对 4 种活动进行了分类。 你可以通过下面的图表来找出，你应该创建哪种类型的活动。 由于安全编程最佳实践根据活动的使用方式而有所不同，因此我们也将解释活动的实现。

表 4-1 活动类型的定义

类型	定义
私有	不能由其他应用加载，所以是最安全的活动
公共	应该由很多未指定的应用使用的活动
伙伴	只能由可信的伙伴公司开发的应用使用的活动
内部	只能由其他内部应用使用的活动

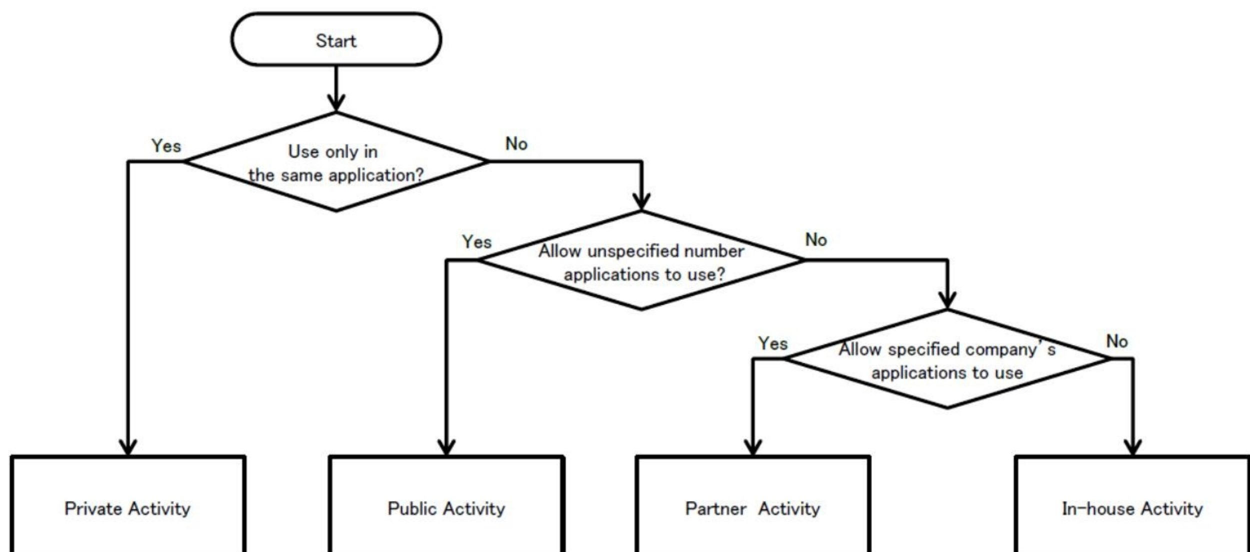


Figure 4.1-1

4.1.1.1 创建/使用私有活动

私有活动是其他应用程序无法启动的活动，因此它是最安全的活动。

当使用仅在应用程序中使用的活动（私有活动）时，只要你对类使用显示意图，那么你不必担心将它意外发送到任何其他应用程序。 但是，第三方应用程序可能会读取用于启动活动的意图。 因此，如果你将敏感信息放入用于启动活动的意图中，有必要采取对策，来确保它不会被恶意第三方读取。

下面展示了如何创建私有活动的示例代码。

要点（创建活动）：

1) 不要指定`taskAffinity`。

2) 不要指定`launchMode`。

3) 将导出属性明确设置为`false`。

4) 仔细和安全地处理收到的意图，即使意图从相同的应用发送。

5) 敏感信息可以发送，因为它发送和接收所有同一应用中的信息。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.activity.privateactivity" >

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >

        <!-- Private activity -->
        <!-- *** POINT 1 *** Do not specify taskAffinity -->
        <!-- *** POINT 2 *** Do not specify launchMode -->
        <!-- *** POINT 3 *** Explicitly set the exported attribu
te to false. -->
        <activity
            android:name=".PrivateActivity"
            android:label="@string/app_name"
            android:exported="false" />

        <!-- Public activity launched by launcher -->
        <activity
            android:name=".PrivateUserActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

PrivateActivity.java

```
package org.jssec.android.activity.privateactivity;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class PrivateActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.private_activity);
        // *** POINT 4 *** Handle the received Intent carefully and securely, even though the Intent was sent from the same application.
        // Omitted, since this is a sample. Please refer to "3.2 Handling Input Data Carefully and Securely."
        String param = getIntent().getStringExtra("PARAM");
        Toast.makeText(this, String.format("Received param: ￥"%s￥"", param), Toast.LENGTH_LONG).show();
    }

    public void onReturnResultClick(View view) {
        // *** POINT 5 *** Sensitive information can be sent since it is sending and receiving all within the same application.
        Intent intent = new Intent();
        intent.putExtra("RESULT", "Sensitive Info");
        setResult(RESULT_OK, intent);
        finish();
    }
}
```

下面展示如何使用私有活动的示例代码。

要点（使用活动）；

6) 不要为意图设置 `FLAG_ACTIVITY_NEW_TASK` 标志来启动活动。

7) 使用显式意图，以及用于调用相同应用中的活动的特定的类。

8) 由于目标活动位于同一个应用中，因此只能通过 `putExtra()` 发送敏感信息[1]。

> 警告：如果不遵守第 1, 2 和 6 点，第三方可能会读到意图。 更多详细信息，请参阅第 4.1.2.2 和 4.1.2.3 节。

9) 即使数据来自同一应用中的活动，也要小心并安全地处理收到的结果数据。

PrivateUserActivity.java

```
package org.jssec.android.activity.privateactivity;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class PrivateUserActivity extends Activity {

    private static final int REQUEST_CODE = 1;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.user_activity);
    }

    public void onUseActivityClick(View view) {
        // *** POINT 6 *** Do not set the FLAG_ACTIVITY_NEW_TASK
 flag for intents to start an activity.
        // *** POINT 7 *** Use the explicit Intents with the cla
ss specified to call an activity in the same application.
        Intent intent = new Intent(this, PrivateActivity.class);
        // *** POINT 8 *** Sensitive information can be sent onl
y by putExtra() since the destination activity is in the same ap
plication.
        intent.putExtra("PARAM", "Sensitive Info");
        startActivityForResult(intent, REQUEST_CODE);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode
, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode != RESULT_OK) return;
        switch (requestCode) {
            case REQUEST_CODE:
                String result = data.getStringExtra("RESULT");
                // *** POINT 9 *** Handle the received data care
fully and securely,
                // even though the data comes from an activity w
ithin the same application.
                // Omitted, since this is a sample. Please refer
 to "3.2 Handling Input Data Carefully and Securely."
                Toast.makeText(this, String.format("Received res
ult: ￥"%s￥"", result), Toast.LENGTH_LONG).show();
                break;
        }
    }
}
```

4.1.1.2 创建/使用公共活动

公共活动是应该由大量未指定的应用程序使用的活动。 有必要注意的是，公共活动可能收到恶意软件发送的意图。 另外，使用公共活动时，有必要注意恶意软件也可以接收或阅读发送给他们的意图。

要点（创建活动）：

1) 将导出属性显式设置为 `true`。

2) 小心并安全地处理接收到的意图。

3) 返回结果时，请勿包含敏感信息。

下面展示了创建公共活动的示例代码。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
    <manifest xmlns:android="http://schemas.android.com/apk/res/
android"
    package="org.jssec.android.activity.publicactivity" >

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >

        <!-- Public Activity -->
        <!-- *** POINT 1 *** Explicitly set the exported attribu
te to true. -->
        <activity
            android:name=".PublicActivity"
            android:label="@string/app_name"
            android:exported="true">

            <!-- Define intent filter to receive an implicit int
ent for a specified action -->
            <intent-filter>
                <action android:name="org.jssec.android.activity
.MY_ACTION" />
                <category android:name="android.intent.category.
DEFAULT" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

PublicActivity.java

```
package org.jssec.android.activity.publicactivity;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class PublicActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        // *** POINT 2 *** Handle the received intent carefully
and securely.
        // Since this is a public activity, it is possible that
the sending application may be malware.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        String param = getIntent().getStringExtra("PARAM");
        Toast.makeText(this, String.format("Received param: ¥"%s¥
"", param), Toast.LENGTH_LONG).show();
    }

    public void onReturnResultClick(View view) {
        // *** POINT 3 *** When returning a result, do not inclu
de sensitive information.
        // Since this is a public activity, it is possible that
the receiving application may be malware.
        // If there is no problem if the data gets received by m
alware, then it can be returned as a result.
        Intent intent = new Intent();
        intent.putExtra("RESULT", "Not Sensitive Info");
        setResult(RESULT_OK, intent);
        finish();
    }
}
```

接下来，这里是公共活动用户端的示例代码。

要点（使用活动）：

4) 不要发送敏感信息。

5) 收到结果时，请仔细并安全地处理数据。

PublicUserActivity.java

```
package org.jssec.android.activity.publicuser;
import android.app.Activity;
import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
public class PublicUserActivity extends Activity {

    private static final int REQUEST_CODE = 1;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void onUseActivityClick(View view) {
        try {
            // *** POINT 4 *** Do not send sensitive information.

            Intent intent = new Intent("org.jssec.android.activi
ty.MY_ACTION");
            intent.putExtra("PARAM", "Not Sensitive Info");
            startActivityForResult(intent, REQUEST_CODE);
        } catch (ActivityNotFoundException e) {
            Toast.makeText(this, "Target activity not found.", T
oast.LENGTH_LONG).show();
        }
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode
, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        // *** POINT 5 *** When receiving a result, handle the d
ata carefully and securely.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        if (resultCode != RESULT_OK) return;
        switch (requestCode) {
            case REQUEST_CODE:
                String result = data.getStringExtra("RESULT");
                Toast.makeText(this, String.format("Received res
ult: ¥"%s¥"", result), Toast.LENGTH_LONG).show();
                break;
        }
    }
}
```

4.1.1.3 创建/使用伙伴活动

伙伴活动是只能由特定应用程序使用的活动。 它们在想要安全共享信息和功能的伙伴公司之间使用。

第三方应用程序可能会读取用于启动活动的意图。 因此，如果你将敏感信息放入用于启动活动的意图中，有必要采取对策来确保其无法被恶意第三方读取。

创建伙伴活动的示例代码如下所示。

要点（创建活动）：

1) 不要指定 `taskAffinity`。

2) 不要指定 `launchMode`。

3) 不要定义意图过滤器，并将导出属性明确设置为 `true`。

4) 通过预定义白名单验证请求应用程序的证书。

5) 尽管意图是从伙伴应用程序发送的，仔细和安全地处理接收到的意图。

6) 只返回公开给伙伴应用的信息。

请参阅“4.1.3.2 验证和请求应用”，了解如何通过白名单验证应用。 此外，请参阅“5.2.1.3 如何验证应用证书的哈希”，了解如何验证白名单中指定目标应用的证书哈希。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:padding="5dp" >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="@string/description" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:onClick="onReturnResultClick"
        android:text="@string/return_result" />
</LinearLayout>
```

PartnerActivity.java

```
package org.jssec.android.activity.partneractivity;

import org.jssec.android.shared.PkgCertWhitelists;
import org.jssec.android.shared.Utils;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class PartnerActivity extends Activity {

    // *** POINT 4 *** Verify the requesting application's certi
ficate through a predefined whitelist.
    private static PkgCertWhitelists sWhitelists = null;

    private static void buildWhitelists(Context context) {
        boolean isdebug = Utils.isDebuggable(context);
        sWhitelists = new PkgCertWhitelists();
        // Register certificate hash value of partner applicatio
n org.jssec.android.activity.partneruser
            .sWhitelists.add("org.jssec.android.activity.partner
user", isdebug ?
            // Certificate hash value of "androiddebugkey" in th
e debug.keystore.
            "0EFB7236 328348A9 89718BAD DF57F544 D5CCB4AE B9DB34
BC 1E29DD26 F77C8255" :
            // Certificate hash value of "partner key" in the ke
ystore.
            "1F039BB5 7861C27A 3916C778 8E78CE00 690B3974 3EB825
9F E2627B8D 4C0EC35A");
            // Register the other partner applications in the sa
me way.
    }

    private static boolean checkPartner(Context context, String 
pkgname) {
        if (sWhitelists == null) buildWhitelists(context);
        return sWhitelists.test(context, pkgname);
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // *** POINT 4 *** Verify the requesting application's c
ertificate through a predefined whitelist.
        if (!checkPartner(this, getCallingActivity().getPackageN
```

```
ame())) {
            Toast.makeText(this,
                "Requesting application is not a partner applica
tion.",
                Toast.LENGTH_LONG).show();
            finish();
            return;
        }
        // *** POINT 5 *** Handle the received intent carefully
and securely, even though the intent was sent from a partner app
lication.
        // Omitted, since this is a sample. Refer to "3.2 Handli
ng Input Data Carefully and Securely."
        Toast.makeText(this, "Accessed by Partner App", Toast.LE
NGTH_LONG).show();
    }

    public void onReturnResultClick(View view) {
        // *** POINT 6 *** Only return Information that is grant
ed to be disclosed to a partner application.
        Intent intent = new Intent();
        intent.putExtra("RESULT", "Information for partner appli
cations");
        setResult(RESULT_OK, intent);
        finish();
    }
}
```

PkgCertWhitelists.java

```
package org.jssec.android.shared;

import java.util.HashMap;
import java.util.Map;
import android.content.Context;

public class PkgCertWhitelists {

    private Map<String, String> mWhitelists = new HashMap<String
, String>();

    public boolean add(String pkgname, String sha256) {
        if (pkgname == null) return false;
        if (sha256 == null) return false;
        sha256 = sha256.replaceAll(" ", "");
        if (sha256.length() != 64) return false; // SHA-256 -> 3
2 bytes -> 64 chars
        sha256 = sha256.toUpperCase();
        if (sha256.replaceAll("[0-9A-F]+", "").length() != 0) re
turn false; // found non hex char
        mWhitelists.put(pkgname, sha256);
        return true;
    }

    public boolean test(Context ctx, String pkgname) {
        // Get the correct hash value which corresponds to pkgna
me.
        String correctHash = mWhitelists.get(pkgname);
        // Compare the actual hash value of pkgname with the cor
rect hash value.
        return PkgCert.test(ctx, pkgname, correctHash);
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;

public class PkgCert {

    public static boolean test(Context ctx, String pkgname, Stri
ng correctHash) {
        if (correctHash == null) return false;
```

```
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
    }
}
```

使用伙伴活动的示例代码如下：

7) 验证目标应用的证书是否已在白名单中注册。

8) 不要为启动活动的意图设置 `FLAG_ACTIVITY_NEW_TASK` 标志。

9) 仅通过 `putExtra()` 发送公开给伙伴活动的信息。

10) 使用显示意图调用伙伴活动。

11) 使用 `startActivityForResult()` 来调用伙伴活动。

12) 即使数据来自伙伴应用程序，也要小心并安全地处理收到的结果数据。

请参阅“4.1.3.2 验证请求应用”了解如何通过白名单验证应用程序。 另请参阅“5.2.1.3 如何验证应用证书的哈希”，了解如何验证白名单中指定目标应用的证书哈希。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
    <manifest xmlns:android="http://schemas.android.com/apk/res/
android"
    package="org.jssec.android.activity.partneruser" >

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >

        <activity
            android:name="org.jssec.android.activity.partneruser
.PartnerUserActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

PartnerUserActivity.java

```
package org.jssec.android.activity.partneruser;

import org.jssec.android.shared.PkgCertWhitelists;
import org.jssec.android.shared.Utils;
import android.app.Activity;
import android.content.ActivityNotFoundException;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class PartnerUserActivity extends Activity {

    // *** POINT 7 *** Verify if the certificate of a target app
lication has been registered in a whitelist.
    private static PkgCertWhitelists sWhitelists = null;
```

```
    private static void buildWhitelists(Context context) {
        boolean isdebug = Utils.isDebuggable(context);
        sWhitelists = new PkgCertWhitelists();
        // Register the certificate hash value of partner applic
ation org.jssec.android.activity.partner
        activity
            .sWhitelists.add("org.jssec.android.activity.partner
activity", isdebug ?
            // The certificate hash value of "androiddebugkey" i
s in debug.keystore.
            "0EFB7236 328348A9 89718BAD DF57F544 D5CCB4AE B9DB34
BC 1E29DD26 F77C8255" :
            // The certificate hash value of "my company key" is
 in the keystore.
            "D397D343 A5CBC10F 4EDDEB7C A10062DE 5690984F 1FB9E8
8B D7B3A7C2 42E142CA");
            // Register the other partner applications in the sa
me way.
    }

    private static boolean checkPartner(Context context, String
pkgname) {
        if (sWhitelists == null) buildWhitelists(context);
        return sWhitelists.test(context, pkgname);
    }

    private static final int REQUEST_CODE = 1;
    // Information related the target partner activity
    private static final String TARGET_PACKAGE = "org.jssec.andr
oid.activity.partneractivity";
    private static final String TARGET_ACTIVITY = "org.jssec.and
roid.activity.partneractivity.PartnerActivity";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void onUseActivityClick(View view) {
        // *** POINT 7 *** Verify if the certificate of the targ
et application has been registered in the own white list.
        if (!checkPartner(this, TARGET_PACKAGE)) {
            Toast.makeText(this, "Target application is not a pa
rtner application.", Toast.LENGTH_LONG).show();
            return;
        }
        try {
            // *** POINT 8 *** Do not set the FLAG_ACTIVITY_NEW_
TASK flag for the intent that start an activity.
            Intent intent = new Intent();
            // *** POINT 9 *** Only send information that is gra
```

```
nted to be disclosed to a Partner Activity only by putExtra().
            intent.putExtra("PARAM", "Info for Partner Apps");
            // *** POINT 10 *** Use explicit intent to call a Pa
rtner Activity.
            intent.setClassName(TARGET_PACKAGE, TARGET_ACTIVITY)
;
            // *** POINT 11 *** Use startActivityForResult() to
call a Partner Activity.
            startActivityForResult(intent, REQUEST_CODE);
        }
        catch (ActivityNotFoundException e) {
            Toast.makeText(this, "Target activity not found.", T
oast.LENGTH_LONG).show();
        }
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode
, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode != RESULT_OK) return;
        switch (requestCode) {
            case REQUEST_CODE:
                String result = data.getStringExtra("RESULT");
                // *** POINT 12 *** Handle the received data car
efully and securely,
                // even though the data comes from a partner app
lication.
                // Omitted, since this is a sample. Please refer
 to "3.2 Handling Input Data Carefully and Securely."
                Toast.makeText(this,
                    String.format("Received result: ¥"%s¥"", res
ult), Toast.LENGTH_LONG).show();
                break;
        }
    }
}
```

PkgCertWhitelists.java

```
package org.jssec.android.shared;

import java.util.HashMap;
import java.util.Map;
import android.content.Context;

public class PkgCertWhitelists {
    private Map<String, String> mWhitelists = new HashMap<String
, String>();

    public boolean add(String pkgname, String sha256) {
        if (pkgname == null) return false;
        if (sha256 == null) return false;
        sha256 = sha256.replaceAll(" ", "");
        if (sha256.length() != 64) return false; // SHA-256 -> 3
2 bytes -> 64 chars
        sha256 = sha256.toUpperCase();
        if (sha256.replaceAll("[0-9A-F]+", "").length() != 0) re
turn false; // found non hex char
        mWhitelists.put(pkgname, sha256);
        return true;
    }

    public boolean test(Context ctx, String pkgname) {
        // Get the correct hash value which corresponds to pkgna
me.
        String correctHash = mWhitelists.get(pkgname);
        // Compare the actual hash value of pkgname with the cor
rect hash value.
        return PkgCert.test(ctx, pkgname, correctHash);
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;

public class PkgCert {
        public static boolean test(Context ctx, String pkgname,
String correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
```

```
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
    }
}
```

4.1.1.4 创建/使用内部活动

内部活动是禁止其他内部应用以外的应用使用的活动。 它们用于内部开发的应用，以便安全地共享信息和功能。

第三方应用可能会读取用于启动活动的意图。 因此，如果你将敏感信息放入用于启动活动的意图中，有必要采取对策来确保它不会被恶意第三方读取。

下面展示了创建内部活动的示例代码。

要点（创建活动）：

1) 定义内部签名权限。

2) 不要指定`taskAffinity`。

3) 不要指定`launchMode`。

4) 需要内部签名权限。

5) 不要定义意图过滤器，并将导出属性显式设为`true`。

6) 确认内部签名权限是由内部应用的。

7) 尽管意图是从内部应用发送的，仔细和安全地处理接收到的意图。

8) 由于请求的应用是内部的，因此可以返回敏感信息。

9) 导出 APK 时，请使用与目标应用相同的开发人员密钥对 APK 进行签名。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.activity.inhouseactivity" >

    <!-- *** POINT 1 *** Define an in-house signature permission
 -->
    <permission
    android:name="org.jssec.android.activity.inhouseactivity.MY_
PERMISSION"
    android:protectionLevel="signature" />

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >

        <!-- In-house Activity -->
        <!-- *** POINT 2 *** Do not specify taskAffinity -->
        <!-- *** POINT 3 *** Do not specify launchMode -->
        <!-- *** POINT 4 *** Require the in-house signature perm
ission -->
        <!-- *** POINT 5 *** Do not define the intent filter and
 explicitly set the exported attribute to
        true -->
        <activity
            android:name="org.jssec.android.activity.inhouseacti
vity.InhouseActivity"
            android:exported="true"
            android:permission="org.jssec.android.activity.inhou
seactivity.MY_PERMISSION" />
    </application>
</manifest>
```

InhouseActivity.java

```
package org.jssec.android.activity.inhouseactivity;

import org.jssec.android.shared.SigPerm;
import org.jssec.android.shared.Utils;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class InhouseActivity extends Activity {

    // In-house Signature Permission
```

```
    private static final String MY_PERMISSION = "org.jssec.andro
id.activity.inhouseactivity.MY_PERMISSION";
    // In-house certificate hash value
    private static String sMyCertHash = null;

    private static String myCertHash(Context context) {
        if (sMyCertHash == null) {
            if (Utils.isDebuggable(context)) {
                // Certificate hash value of "androiddebugkey" i
n the debug.keystore.
                sMyCertHash = "0EFB7236 328348A9 89718BAD DF57F5
44 D5CCB4AE B9DB34BC 1E29DD26 F77C8255";
            } else {
                // Certificate hash value of "my company key" in
 the keystore.
                sMyCertHash = "D397D343 A5CBC10F 4EDDEB7C A10062
DE 5690984F 1FB9E88B D7B3A7C2 42E142CA";
            }
        }
        return sMyCertHash;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        // *** POINT 6 *** Verify that the in-house signature pe
rmission is defined by an in-house application.
        if (!SigPerm.test(this, MY_PERMISSION, myCertHash(this))
) {
            Toast.makeText(this, "The in-house signature permiss
ion is not declared by in-house application.",
            Toast.LENGTH_LONG).show();
            finish();
            return;
        }
        // *** POINT 7 *** Handle the received intent carefully
and securely, even though the intent was sent from an in-house a
pplication.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        String param = getIntent().getStringExtra("PARAM");
        Toast.makeText(this, String.format("Received param: ¥"%s¥
"", param), Toast.LENGTH_LONG).show();
    }

    public void onReturnResultClick(View view) {
        // *** POINT 8 *** Sensitive information can be returned
 since the requesting application is inhouse.
        Intent intent = new Intent();
        intent.putExtra("RESULT", "Sensitive Info");
        setResult(RESULT_OK, intent);
        finish();
```

```
    }
}
```

SigPerm.java

```
package org.jssec.android.shared;

import android.content.Context;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.PermissionInfo;

public class SigPerm {

    public static boolean test(Context ctx, String sigPermName,
String correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, sigPermName));
    }

    public static String hash(Context ctx, String sigPermName) {
        if (sigPermName == null) return null;
        try {
            // Get the package name of the application which dec
lares a permission named sigPermName.
            PackageManager pm = ctx.getPackageManager();
            PermissionInfo pi;
            pi = pm.getPermissionInfo(sigPermName, PackageManage
r.GET_META_DATA);
            String pkgname = pi.packageName;
            // Fail if the permission named sigPermName is not a
 Signature Permission
            if (pi.protectionLevel != PermissionInfo.PROTECTION_
SIGNATURE) return null;
            // Return the certificate hash value of the applicat
ion which declares a permission named sigPermName.
            return PkgCert.hash(ctx, pkgname);
        } catch (NameNotFoundException e) {
            return null;
        }
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
```

```
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;

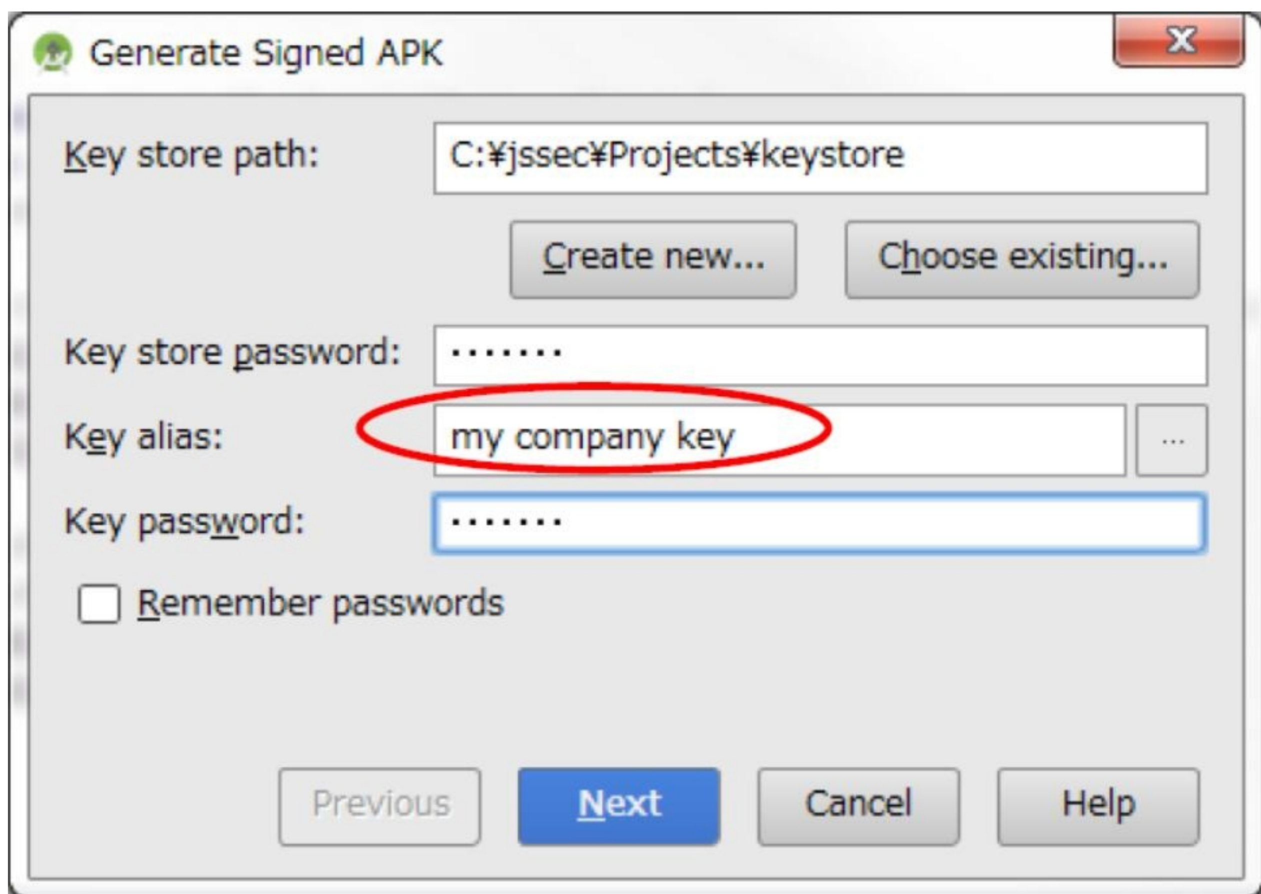
public class PkgCert {
    public static boolean test(Context ctx, String pkgname, Stri
ng correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
    }
}
```

要点 9：导出 APK 时，请使用与目标应用相同的开发人员密钥对 APK 进行签名。



使用内部活动的代码如下：

要点（使用活动）：

10) 声明你要使用内部签名权限。

11) 确认内部签名权限是由内部应用定义的。

12) 验证目标应用是否使用内部证书签名。

13) 由于目标应用是内部的，所以敏感信息只能由 `putExtra()` 发送。

14) 使用显式意图调用内部活动。

15) 即使数据来自内部应用，也要小心并安全地处理接收到的数据。

16) 导出 APK 时，请使用与目标应用相同的开发人员密钥对 APK 进行签名。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.activity.inhouseuser" >

    <!-- *** POINT 10 *** Declare to use the in-house signature
permission -->
    <uses-permission
    android:name="org.jssec.android.activity.inhouseactivity.MY_
PERMISSION" />

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >

        <activity
            android:name="org.jssec.android.activity.inhouseuser
.InhouseUserActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

InhouseUserActivity.java

```
package org.jssec.android.activity.inhouseuser;

import org.jssec.android.shared.PkgCert;
import org.jssec.android.shared.SigPerm;
import org.jssec.android.shared.Utils;
import android.app.Activity;
import android.content.ActivityNotFoundException;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class InhouseUserActivity extends Activity {

    // Target Activity information
    private static final String TARGET_PACKAGE = "org.jssec.andr
```

```
oid.activity.inhouseactivity";
    private static final String TARGET_ACTIVITY = "org.jssec.and
roid.activity.inhouseactivity.InhouseActivity";
    // In-house Signature Permission
    private static final String MY_PERMISSION = "org.jssec.andro
id.activity.inhouseactivity.MY_PERMISSION";
    // In-house certificate hash value
    private static String sMyCertHash = null;

    private static String myCertHash(Context context) {
        if (sMyCertHash == null) {
            if (Utils.isDebuggable(context)) {
                // Certificate hash value of "androiddebugkey" i
n the debug.keystore.
                sMyCertHash = "0EFB7236 328348A9 89718BAD DF57F5
44 D5CCB4AE B9DB34BC 1E29DD26 F77C8255";
            } else {
                // Certificate hash value of "my company key" in
 the keystore.
                sMyCertHash = "D397D343 A5CBC10F 4EDDEB7C A10062
DE 5690984F 1FB9E88B D7B3A7C2 42E142CA";
            }
        }
        return sMyCertHash;
    }

    private static final int REQUEST_CODE = 1;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void onUseActivityClick(View view) {
        // *** POINT 11 *** Verify that the in-house signature p
ermission is defined by an in-house application.
        if (!SigPerm.test(this, MY_PERMISSION, myCertHash(this))
) {
            Toast.makeText(this, "The in-house signature permiss
ion is not declared by in-house application.",
            Toast.LENGTH_LONG).show();
            return;
        }
        // ** POINT 12 *** Verify that the destination applicati
on is signed with the in-house certificate.
        if (!PkgCert.test(this, TARGET_PACKAGE, myCertHash(this)
)) {
            Toast.makeText(this, "Target application is not an i
n-house application.", Toast.LENGTH_LONG).show();
            return;
        }
        try {
```

```
            Intent intent = new Intent();
            // *** POINT 13 *** Sensitive information can be sen
t only by putExtra() since the destination application is in-hou
se.
            intent.putExtra("PARAM", "Sensitive Info");

            // *** POINT 14 *** Use explicit intents to call an
In-house Activity.
            intent.setClassName(TARGET_PACKAGE, TARGET_ACTIVITY)
;
            startActivityForResult(intent, REQUEST_CODE);
        } catch (ActivityNotFoundException e) {
            Toast.makeText(this, "Target activity not found.", T
oast.LENGTH_LONG).show();
        }
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode
, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode != RESULT_OK) return;
        switch (requestCode) {
            case REQUEST_CODE:
                String result = data.getStringExtra("RESULT");
                // *** POINT 15 *** Handle the received data car
efully and securely,
                // even though the data came from an in-house ap
plication.
                // Omitted, since this is a sample. Please refer
 to "3.2 Handling Input Data Carefully and Securely."
                Toast.makeText(this, String.format("Received res
ult: ¥"%s¥"", result), Toast.LENGTH_LONG).show();
                break;
        }
    }
}
```

SigPerm.java

```
package org.jssec.android.shared;

import android.content.Context;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.PermissionInfo;

public class SigPerm {

    public static boolean test(Context ctx, String sigPermName,
String correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, sigPermName));
    }

    public static String hash(Context ctx, String sigPermName) {
        if (sigPermName == null) return null;
        try {
            // Get the package name of the application which dec
lares a permission named sigPermName.
            PackageManager pm = ctx.getPackageManager();
            PermissionInfo pi;
            pi = pm.getPermissionInfo(sigPermName, PackageManage
r.GET_META_DATA);
            String pkgname = pi.packageName;
            // Fail if the permission named sigPermName is not a
 Signature Permission
            if (pi.protectionLevel != PermissionInfo.PROTECTION_
SIGNATURE) return null;
            // Return the certificate hash value of the applicat
ion which declares a permission named sigPermName.
            return PkgCert.hash(ctx, pkgname);
        } catch (NameNotFoundException e) {
            return null;
        }
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;
```

```java
public class PkgCert {

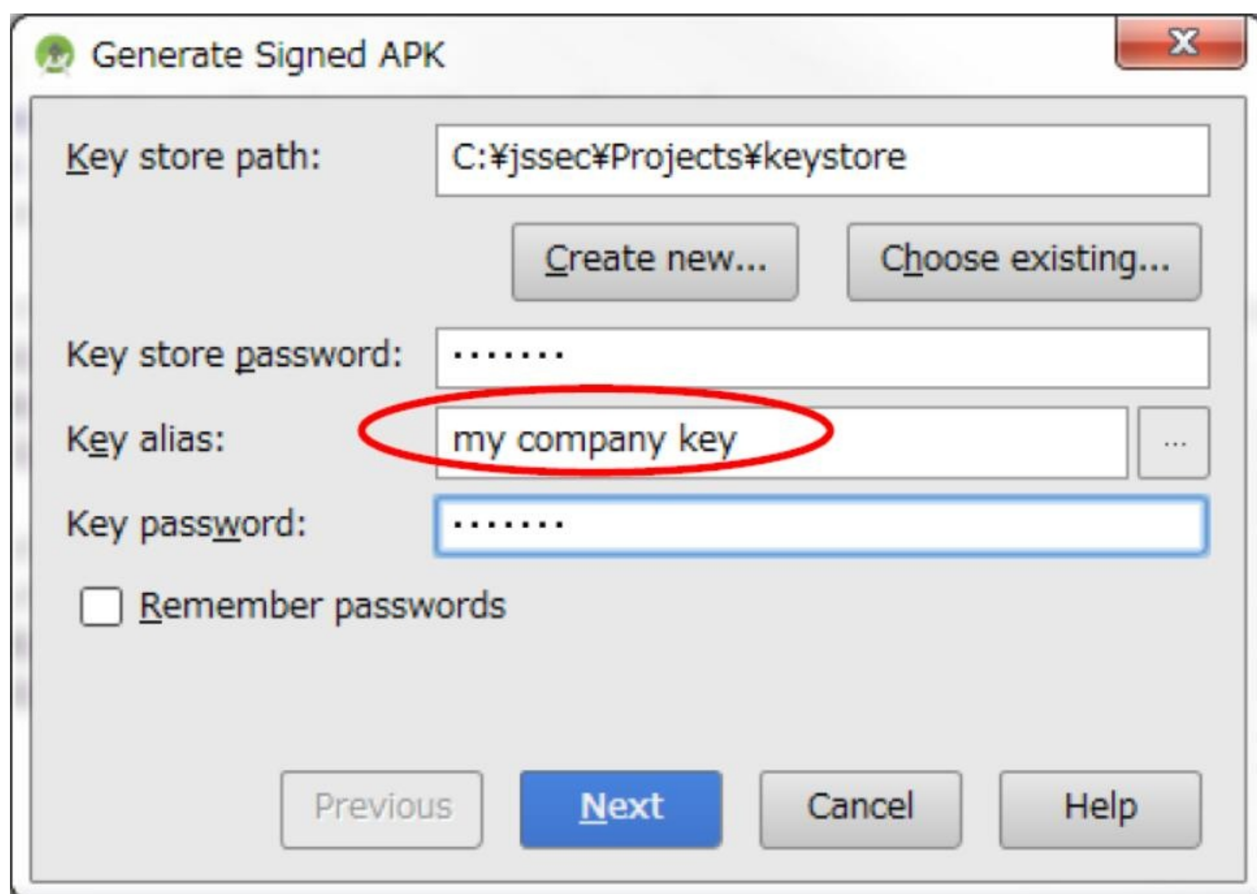
    public static boolean test(Context ctx, String pkgname, Stri
ng correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
    }
}
```

要点 16：导出 APK 时，请使用与目标应用相同的开发人员密钥对 APK 进行签名。


Generate Signed APK
Key store path:
C:¥jssec¥Projects¥keystore
Create new...
Choose existing...
Key store password:
Key alias:
my company key
Key password:
Remember passwords
Previous
Next
Cancel
Help

4.1.2 规则书

创建或向活动发送意图时，请务必遵循以下规则。

4.1.2.1 仅在应用内部使用的活动必须设置为私有（必需）

仅在单个应用中使用的活动，不需要能够从其他应用接收任何意图。 开发人员经常假设，应该是私有的活动不会受到攻击，但有必要将这些活动显式设置为私有，以阻止恶意内容被收到。

AndroidManifest.xml

```
<!-- Private activity -->
<!-- *** POINT 3 *** Explicitly set the exported attribute to fa
lse. -->
<activity
    android:name=".PrivateActivity"
    android:label="@string/app_name"
    android:exported="false" />
```

意图过滤器不应该设置在仅用于单个应用的活动中。 由于意图过滤器的特性，以及工作原理，即使您打算向内部的私有活动发送意图，但如果通过意图过滤器发送，则可能会无意中启动另一个活动。 更多详细信息，请参阅高级主题“4.1.3.1 结合导出属性和意图过滤器设置（用于活动）”。

AndroidManifest.xml（不推荐）

```
<!-- Private activity -->
<!-- *** POINT 3 *** Explicitly set the exported attribute to fa
lse. -->
<activity
    android:name=".PictureActivity"
    android:label="@string/picture_name"
    android:exported="false" >
    <intent-filter>
        <action android:name="org.jssec.android.activity.OPEN />
    </intent-filter>
</activity>
```

4.1.2.2 不要指定 `taskAffinity`（必需）

在 Android OS 中，活动由任务管理。 任务名称由根活动所具有的 Affinity 决定。 另一方面，对于根活动以外的活动，活动所属的任务不仅仅取决于 Affinity，还取决于活动的启动模式。 更多详细信息，请参阅“4.1.3.4 根活动”。

在默认设置中，每个活动使用其包名称作为其 Affinity。 因此，任务根据应用分配，因此单个应用中的所有活动都属于同一个任务。 要更改任务分配，您可以在 `AndroidManifest.xml` 文件中显式声明 Affinity，或者您可以在发送给活动的意图中，设置一个标志。 但是，如果更改任务分配，则存在风险，即其他应用可能读取一些意图，它发送给属于其他任务的活动。

请务必不要在 `AndroidManifest.xml` 文件中指定 `android:taskAffinity` ，并使用默认设置，将 affinity 作为包名，以防止其他应用读取发送或接收的意图中的敏感信息。

以下是用于创建和使用私有活动的 `AndroidManifest.xml` 示例文件。

AndroidManifest.xml

```
<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name" >
    <!-- Private activity -->
    <!-- *** POINT 1 *** Do not specify taskAffinity -->
    <activity
        android:name=".PrivateActivity"
        android:label="@string/app_name"
        android:exported="false" />
</application>
```

任务和 Affinity 的更多信息，请参阅“Google Android 编程指南” [2]，Google 开发者 API 指南“任务和返回栈” [3]，“4.1.3.3 读取发送到活动的意图”和“4.1.3.4 根活动”

> [2] Author Egawa, Fujii, Asano, Fujita, Yamada, Yamaoka, Sano, Takebata, “Google Android Programming Guide”, ASCII Media Works, July 2009
>
> [3] http://developer.android.com/guide/components/tasks-and-back-stack.html

4.1.2.3 不要指定 `launchMode` （必需）

活动的启动模式，用于控制启动活动时的设置，它用于创建新任务和活动实例。 默认情况下，它被设置为 `"standard"` 。 在 `"standard"` 设置中，新实例总是在启动活动时创建，任务遵循属于调用活动的任务，并且不可能创建新任务。 创建新任务时，其他应用可能会读取调用意图的内容，因此当敏感信息包含在意图中时，需要使用 `"standard"` 活动启动模式设置。 活动的启动模式可以在 `AndroidManifest.xml` 文件的 `android:launchMode` 属性中显式设置，但由于上面解释的原因，这不应该在活动的声明中设置，并且该值应该保留为默认的 `"standard"` 。

AndroidManifest.xml

```
<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name" >
    <!-- Private activity -->
    <!-- *** POINT 2 *** Do not specify launchMode -->
    <activity
        android:name=".PrivateActivity"
        android:label="@string/app_name"
        android:exported="false" />
</application>
```

请参阅“4.1.3.3 读取发送到活动的意图”和“4.1.3.4 根活动”。

4.1.2.4 不要为启动活动的意图设置 `FLAG_ACTIVITY_NEW_TASK` 标志（必需）

执行 `startActivity()` 或 `startActivityForResult()` 时，可以更改 `Activity` 的启动模式，并且在某些情况下可能会生成新任务。 因此有必要在执行期间不更改 `Activity` 的启动模式。

要更改 `Activity` 启动模式，使用 `setFlags()` 或 `addFlags()` 设置 `Intent` 标志，并将该 `Intent` 用作 `startActivity()` 或 `startActivityForResult()` 的参数。 `FLAG_ACTIVITY_NEW_TASK` 是用于创建新任务的标志。 当设置 `FLAG_ACTIVITY_NEW_TASK` 时，如果被调用的 `Activity` 不存在于后台或前台，则会创建一个新任务。 `FLAG_ACTIVITY_MULTIPLE_TASK` 标志可以与 `FLAG_ACTIVITY_NEW_TASK` 同时设置。 在这种情况下，总会创建一个新的任务。 新任务可以通过任一设置创建，因此不应使用处理敏感信息的意图来设置这些东西。

```
// *** POINT 6 *** Do not set the FLAG_ACTIVITY_NEW_TASK flag fo
r the intent to start an activity.
Intent intent = new Intent(this, PrivateActivity.class);
intent.putExtra("PARAM", "Sensitive Info");
startActivityForResult(intent, REQUEST_CODE);
```

另外，即使通过明确设置 `FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS` 标志创建了新任务，您也可能认为有一种方法可以防止读取 `Intent` 的内容。 但是，即使使用此方法，内容也可以由第三方读取，因此您应该避免使用 `FLAG_ACTIVITY_NEW_TASK` 。

请参阅“4.1.3.1 结合导出属性和意图过滤设置（针对活动）”，“4.1.3.3 读取发送到活动的意图”和“4.1.3.4 根活动”。

4.1.2.5 小心和安全地处理收到的意图

风险因 `Activity` 的类型而异，但在处理收到的 `Intent` 数据时，您应该做的第一件事是输入验证。

由于公共活动可以从不受信任的来源接收意图，它们可能会受到恶意软件的攻击。另一方面，私有活动永远不会直接从其他应用收到任何意图，但目标应用中的公共活动可能会将恶意 `Intent` 转发给私有活动，因此您不应该认为私有活动不会收到任何恶意输入。 由于伙伴活动和内部活动也有恶意意图转发给他们的风险，因此有必要对这些意图进行输入验证。

请参阅“3.2 仔细和安全地处理输入数据”

4.1.2.6 在验证签名权限由内部应用定义之后，使用内部定义的签名权限（必需）

确保在创建活动时，通过定义内部签名权限来保护您的内部活动。 由于在 `AndroidManifest.xml` 文件中定义权限或声明权限请求不能提供足够的安全性，请务必参考“5.2.1.2 如何使用内部定义的签名权限，在内部应用之间进行通信”。

4.1.2.7 返回结果时，请注意目标应用产生的可能的信息泄露（必需）

当您使用 `setResult()` 返回数据时，目标应用的可靠性将取决于 `Activity` 类型。 当公共活动用于返回数据时，目标可能会成为恶意软件，在这种情况下，可能会以恶意方式使用该信息。 对于私有和内部活动，不需要过多担心返回的数据被恶意使用，因为它们被返回到您控制的应用。 伙伴活动中间有些东西。

如上所述，当从活动中返回数据时，您需要注意来自目标应用的信息泄漏。

```
public void onReturnResultClick(View view) {
    // *** POINT 6 *** Information that is granted to be disclos
ed to a partner application can be returned.
    Intent intent = new Intent();
    intent.putExtra("RESULT", "Sensitive Info");
    setResult(RESULT_OK, intent);
    finish();
}
```

4.1.2.8 如果目标活动是预先确定的，则使用显式意图（必需）

当通过隐式意图使用 `Activity` 时， `Intent` 发送到的 `Activity` 由 Android OS 确定。 如果意图被错误地发送到恶意软件，则可能发生信息泄漏。 另一方面，当通过显式意图使用 `Activity` 时，只有预期的 `Activity` 会收到 `Intent` ，所以这样更安全。 除非用户需要确定意图应该发送到哪个应用活动，否则应该使用显式意图并提前指定目标。

```
Intent intent = new Intent(this, PictureActivity.class);
intent.putExtra("BARCODE", barcode);
startActivity(intent);
```

```
Intent intent = new Intent();
intent.setClassName(
    "org.jssec.android.activity.publicactivity",
    "org.jssec.android.activity.publicactivity.PublicActivity");
startActivity(intent);
```

但是，即使通过显式意图使用其他应用的公共活动，目标活动也可能是恶意软件。这是因为，即使通过软件包名称限制目标，恶意应用仍可能伪造与真实应用相同的软件包名称。为了消除这种风险，有必要考虑使用伙伴或内部活动。

请参阅“4.1.3.1 组合导出属性和意图过滤器设置（对于活动）”

4.1.2.9 小心并安全地处理来自被请求活动的返回数据（必需）

根据您访问的活动类型，风险略有不同，但在处理作为返回值的收到的 `Intent` 数据，您始终需要对接收到的数据执行输入验证。公共活动必须接受来自不受信任来源的返回意图，因此在访问公共活动时，返回的意图实际上可能是由恶意软件发送的。人们往往错误地认为，私有活动返回的所有内容都是安全的，因为它们来源于同一个应用。但是，由于从不可信来源收到的意图可能会间接转发，因此您不应盲目信任该意图的内容。伙伴和内部活动在私有和公共活动中间有一定风险。一定也要对这些活动输入验证。更多信息，请参阅“3.2 仔细和安全地处理输入数据”。

4.1.2.10 如果与其他公司的应用链接，请验证目标活动（必需）

与其他公司的应用链接时，确保确定了白名单。您可以通过在应用内保存公司的证书散列副本，并使用目标应用的证书散列来检查它。这将防止恶意应用欺骗意图。具体实现方法请参考示例代码“4.1.1.3 创建/使用伙伴活动”部分。技术细节请参阅“4.1.3.2 验证请求应用”。

4.2.11 提供二手素材时，素材应受到同等保护（必需）

当受到权限保护的信息或功能素材被另一个应用提供时，您需要确保它具有访问素材所需的相同权限。在 Android OS 权限安全模型中，只有已获得适当权限的应用才可以直接访问受保护的素材。但是，存在一个漏洞，因为具有素材权限的应用可以充当代理，并允许非特权应用程序访问它。基本上这与重新授权相同，因此它被称为“重新授权”问题。请参阅“5.2.3.4 重新授权问题”。

4.2.12 敏感信息的发送应该尽可能限制（推荐）

您不应将敏感信息发送给不受信任的各方。 即使您正在连接特定的应用程序，仍有可能无意中将 `Intent` 发送给其他应用程序，或者恶意第三方可能会窃取您的意图。 请参阅“4.1.3.5 使用活动时的日志输出”。

将敏感信息发送到活动时，您需要考虑信息泄露的风险。 您必须假设，发送到公共活动的 `Intent` 中的所有数据都可以由恶意第三方获取。 此外，根据实现，向 `伙伴或内部活动发送意图时，也存在各种信息泄漏的风险。 即使将数据发送到私有活动，LogCat` 泄漏。 意图附加部分中的信息不会输出到 `LogCat`，因此最好在那里存储敏感信息。

但是，不首先发送敏感数据，是防止信息泄露的唯一完美解决方案，因此您应该尽可能限制发送的敏感信息的数量。 当有必要发送敏感信息时，最好的做法是只发送给受信任的活动，并确保信息不能通过 `LogCat` 泄露。

另外，敏感信息不应该发送到根活动。 根活动是创建任务时首先调用的活动。 例如，从启动器启动的活动始终是根活动。

根活动的更多详细信息，请参阅“4.1.3.3 发送到活动的意图”和“4.1.3.4 根活动”。

4.1.3 高级话题

4.1.3.1 组合导出属性和意图过滤器（对于活动）

我们已经解释了如何实现本指南中的四类活动：私有活动，公共活动，伙伴活动和内部活动。 下表中定义了每种类型的导出属性的允许的设置，和 `intent-filter` 元素的各种组合，它们在 `AndroidManifest.xml` 文件中定义。 请使用你尝试创建的活动，验证导出属性和 `intent-filter` 元素的兼容性。

	导出属性的值		
	True	False	未指定
意图过滤器已定义	公开	（不使用）	（不使用）
意图过滤器未定义	公开、伙伴、内部	`AndroidManifest.xml`	（不使用）

表 4.1-2

当未指定 `Activity` 的导出属性时， `Activity` 是否为公开的，取决于 `Activity` 的意图过滤器的存在与否 [4]。 但是，在本手册中，禁止将导出属性设置为未指定。通常，如前所述，最好避免依赖任何给定 API 的默认行为的实现；此外，如果存在明确的方法（例如导出属性）来启用重要的安全相关设置，那么使用这些方法总是一个好主意。

> 如果定义了任何意图过滤器，则该活动是公开的；否则它是私有的。更多信息请参阅 https://developer.android.com/guide/topics/manifest/activity-element.html#exported。

不应该使用未定义的意图过滤器和导出属性 `false` 的原因，是 Android 的行为存在漏洞，并且由于意图过滤器的工作原理，其他应用的活动可能会意外调用它。下面的两个图展示了这个解释。图 4.1-4 是一个正常行为的例子，其中私有活动（应用 A）只能由同一个应用的隐式 `Intent` 调用。 意图过滤器（ `action ="X"` ）被定义为仅在应用 A 内部工作，所以这是预期的行为。

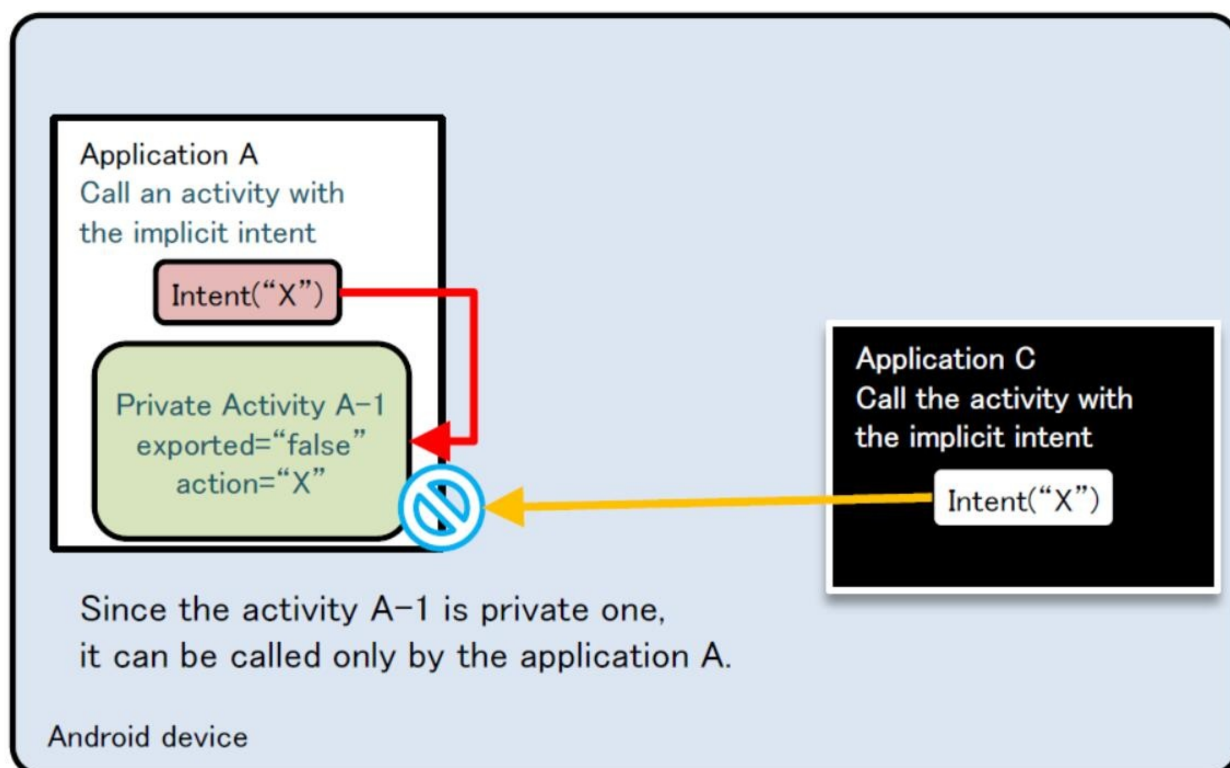


Figure 4.1-4

下面的图 4.1-5 展示了一个场景，其中在应用 B 和应用 A 中定义了相同的意图过滤器（ `action ="X"` ）。应用 A 试图通过发送隐式意图，来调用同一应用中的私有活动 ，但是这次显示了对话框，询问用户选择哪个应用，以及应用 B 中的公共活动 B-1 ，由于用户的选择而错误调用。 由于这个漏洞，可能会将敏感信息发送到其他应用，或者应用可能会收到意外的返回值。

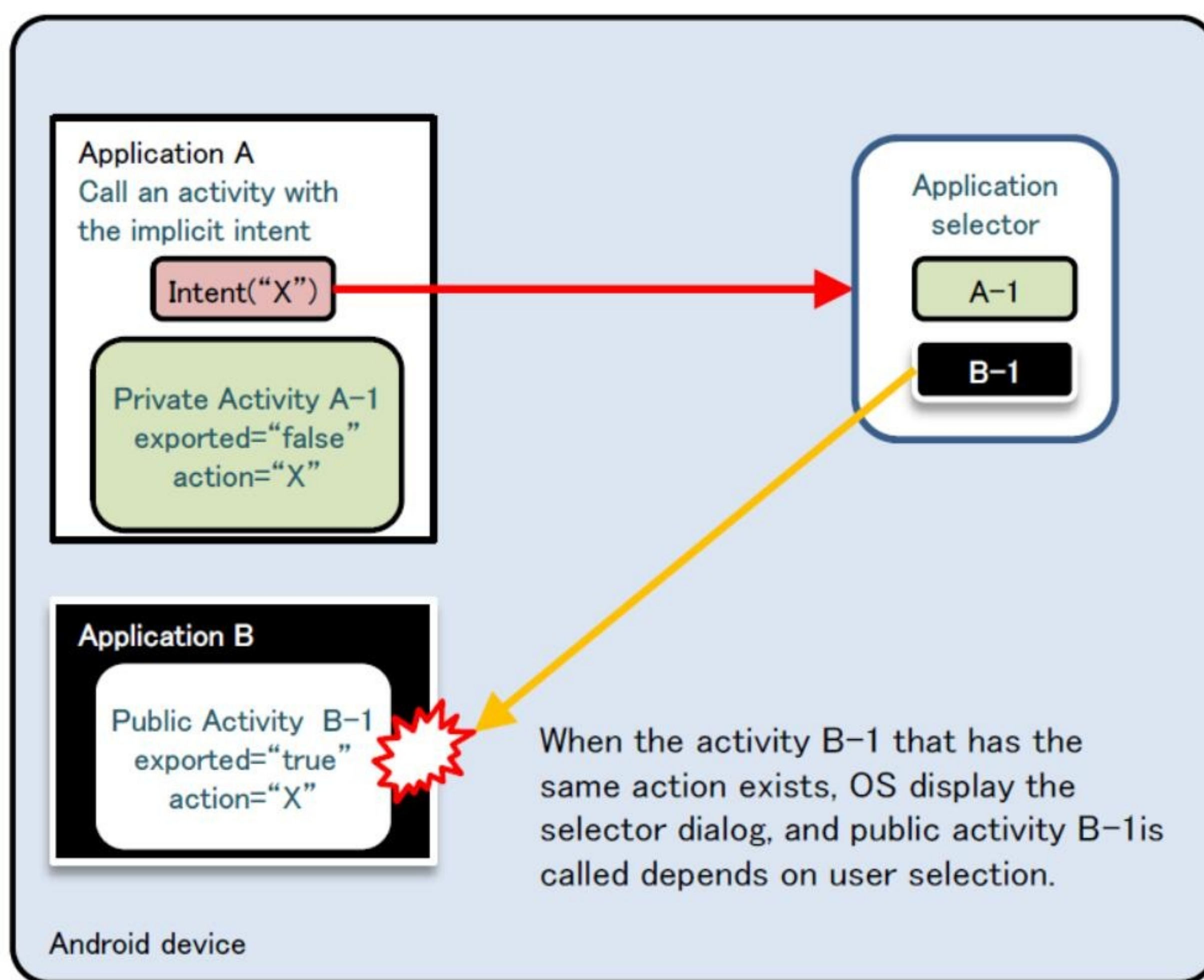


Figure 4.1-5

如上所示，使用意图过滤器，将隐式意图发送到私有应用，可能会导致意外行为，因此最好避免此设置。 另外，我们已经验证了这种行为不依赖于应用 A 和应用 B 的安装顺序。

4.1.3.2 验证请求应用

我们在此解释一些技术信息，关于如何实现伙伴活动。 伙伴应用只允许白名单中注册的特定应用访问，并且所有其他应用都被拒绝。 由于除内部应用之外的其他应用也需要访问权限，因此我们无法使用签名权限进行访问控制。

简而言之，我们希望验证尝试使用伙伴活动的应用，通过检查它是否在预定义的白名单中注册，如果是，则允许访问，如果不是，则拒绝访问。 应用验证的方式是，从请求访问的应用获取证书，并将其与白名单中的散列进行比较。

一些开发人员可能会认为，仅仅比较软件包名称而不获取证书就足够了，但是，很容易伪装成合法应用的软件包名称，因此这不是检查真实性的好方法。 任意指定的值不应用于认证。 另一方面，由于只有应用开发人员拥有用于签署证书的开发人员密钥，因此这是识别的更好方法。 由于证书不容易被伪造，除非恶意第三方可以窃取开发人员密钥，否则恶意应用被信任的可能性很小。 虽然可以将整个证书存储在白名单中，但为了使文件大小最小，仅存储 SHA-256 散列值就足够了。

使用这个方法有两个限制：

- 请求应用需要使用 `startActivityForResult()` 而不是 `startActivity()` 。
- 请求应用应该只从 `Activity` 调用。

第二个限制是由于第一个限制而施加的限制，因此技术上只有一个限制。

由于 `Activity.getCallingPackage()` 的限制，它获取调用应用的包名称，所以会发生此限制。 `Activity.getCallingPackage()` 仅在由 `startActivityForResult()` 调用时，才返回源（请求）应用的包名，但不幸的是，当它由 `startActivity()` 调用时，它仅返回 `null` 。因此，使用此处解释的方法时，源（请求）应用需要使用 `startActivityForResult()` ，即使它不需要获取返回值。另外， `startActivityForResult()` 只能在 `Activity` 类中使用，所以源（请求者）仅限于活动。

PartnerActivity.java

```
package org.jssec.android.activity.partneractivity;

import org.jssec.android.shared.PkgCertWhitelists;
import org.jssec.android.shared.Utils;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class PartnerActivity extends Activity {

    // *** POINT 4 *** Verify the requesting application's certi
ficate through a predefined whitelist.
    private static PkgCertWhitelists sWhitelists = null;

    private static void buildWhitelists(Context context) {
        boolean isdebug = Utils.isDebuggable(context);
        sWhitelists = new PkgCertWhitelists();
        // Register certificate hash value of partner applicatio
n org.jssec.android.activity.partneruser
            .sWhitelists.add("org.jssec.android.activity.partner
user", isdebug ?
            // Certificate hash value of "androiddebugkey" in th
e debug.keystore.
            "0EFB7236 328348A9 89718BAD DF57F544 D5CCB4AE B9DB34
BC 1E29DD26 F77C8255" :
            // Certificate hash value of "partner key" in the ke
ystore.
            "1F039BB5 7861C27A 3916C778 8E78CE00 690B3974 3EB825
9F E2627B8D 4C0EC35A");
            // Register the other partner applications in the sa
```

```
me way.
    }

    private static boolean checkPartner(Context context, String
pkgname) {
        if (sWhitelists == null) buildWhitelists(context);
        return sWhitelists.test(context, pkgname);
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        // *** POINT 4 *** Verify the requesting application's c
ertificate through a predefined whitelist.
        if (!checkPartner(this, getCallingActivity().getPackageN
ame())) {
            Toast.makeText(this,
            "Requesting application is not a partner application
.", Toast.LENGTH_LONG).show();
            finish();
            return;
        }
        // *** POINT 5 *** Handle the received intent carefully
and securely, even though the intent was sent from a partner app
lication.
        // Omitted, since this is a sample. Refer to "3.2 Handli
ng Input Data Carefully and Securely."
        Toast.makeText(this, "Accessed by Partner App", Toast.LE
NGTH_LONG).show();
    }

    public void onReturnResultClick(View view) {
        // *** POINT 6 *** Only return Information that is grant
ed to be disclosed to a partner application.
        Intent intent = new Intent();
        intent.putExtra("RESULT", "Information for partner appli
cations");
        setResult(RESULT_OK, intent);
        finish();
    }
}
```

PkgCertWhitelists.java

```
package org.jssec.android.shared;

import java.util.HashMap;
import java.util.Map;
import android.content.Context;

public class PkgCertWhitelists {

    private Map<String, String> mWhitelists = new HashMap<String
, String>();
    public boolean add(String pkgname, String sha256) {
        if (pkgname == null) return false;
        if (sha256 == null) return false;
        sha256 = sha256.replaceAll(" ", "");
        if (sha256.length() != 64) return false; // SHA-256 -> 3
2 bytes -> 64 chars
        sha256 = sha256.toUpperCase();
        if (sha256.replaceAll("[0-9A-F]+", "").length() != 0) re
turn false; // found non hex char
        mWhitelists.put(pkgname, sha256);
        return true;
    }

    public boolean test(Context ctx, String pkgname) {
        // Get the correct hash value which corresponds to pkgna
me.
        String correctHash = mWhitelists.get(pkgname);
        // Compare the actual hash value of pkgname with the cor
rect hash value.
        return PkgCert.test(ctx, pkgname, correctHash);
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;

public class PkgCert {

    public static boolean test(Context ctx, String pkgname, Stri
ng correctHash) {
        if (correctHash == null) return false;
```

```
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
    }
}
```

4.1.3.3 读取发送给活动的意图

在 Android 5.0（API Level 21）及更高版本中，使用 `getRecentTasks()` 得到的信息仅限于调用者自己的任务，并且可能还有一些其他任务，例如已知不敏感的其他任务。 但是支持 Android 5.0（API Level 21）版本的应用应该防止泄露敏感信息。 以下描述了问题内容，它出现在 Android 5.0 及更早版本中。

发送到任务的根 `Activity` 的意图，被添加到任务历史中。 根活动是在任务中启动的第一个活动。 任何应用都可以通过使用 `ActivityManager` 类，读取添加到任务历史的意图。

下面显示了从应用中读取任务历史的示例代码。 要浏览任务历史，请在 `AndroidManifest.xml` 文件中指定 `GET_TASKS` 权限。

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.intent.maliciousactivity" >

    <!-- Use GET_TASKS Permission -->
    <uses-permission android:name="android.permission.GET_TASKS"
 />

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MaliciousActivity"
            android:label="@string/title_activity_main"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

MaliciousActivity.java

```java
package org.jssec.android.intent.maliciousactivity;

import java.util.List;
import java.util.Set;
import android.app.Activity;
import android.app.ActivityManager;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;

public class MaliciousActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.malicious_activity);
        // Get am ActivityManager instance.
        ActivityManager activityManager = (ActivityManager) getS
ystemService(ACTIVITY_SERVICE);
        // Get 100 recent task info.
        List<ActivityManager.RecentTaskInfo> list = activityMana
ger
            .getRecentTasks(100, ActivityManager.RECENT_WITH_EXC
LUDED);
        for (ActivityManager.RecentTaskInfo r : list) {
            // Get Intent sent to root Activity and Log it.
            Intent intent = r.baseIntent;
            Log.v("baseIntent", intent.toString());
            Log.v(" action:", intent.getAction());
            Log.v(" data:", intent.getDataString());
            if (r.origActivity != null) {
                Log.v(" pkg:", r.origActivity.getPackageName() +
 r.origActivity.getClassName());
            }
            Bundle extras = intent.getExtras();
            if (extras != null) {
                Set<String> keys = extras.keySet();
                for(String key : keys) {
                    Log.v(" extras:", key + "=" + extras.get(key
).toString());
                }
            }
        }
    }
}
```

你可以使用 `AcitivityManager` 类的 `getRecentTasks()` 函数，来获取任务历史的指定条目。 每个任务的信息存储在 `ActivityManager.RecentTaskInfo` 类的实例中，但发送到任务根 `Activity` 的意图存储在其成员变量 `baseIntent` 中。 由

于根 `Activity` 是创建任务时启动的 `Activity` ，请务必在调用 `Activity` 时，不要满足以下两个条件。

- 新的任务在活动被调用时创建
- 被调用的活动是任务的根活动，它已经在前台或者后台存在

4.1.3.4 根活动

根活动是作为任务起点的活动。 换句话说，这是创建任务时启动的活动。 例如，当默认活动由启动器启动时，此活动将是根活动。 根据 Android 规范，发送到根 `Activity` 的意图的内容可以从任意应用中读取。 因此，有必要采取对策，不要将敏感信息发送到根活动。 在本指南中，已经制定了以下三条规则来避免被调用的 `Activity` 成为根活动。

- 不要指定 `taskAffinity`
- 不要指定 `launchMode`
- 发送给活动的意图中，不要设置 `FLAG_ACTIVITY_NEW_TASK`

我们考虑一个情况，活动可以成为下面的根活动。 被调用的活动成为根活动，取决于以下内容。

- 被调用活动的启动模式
- 被调用活动的任务及其启动模式

首先，让我解释一下“被调用活动的启动模式”。 可以通过在 `AndroidManifest.xml` 中编写 `android:launchMode` 来设置 `Activity` 的启动模式。 当它没有编写时，它被认为是“标准”。 另外，启动模式也可以通过设置意图的标志来更改。 标志 `FLAG_ACTIVITY_NEW_TASK` 以 `singleTask` 模式启动活动。

启动模式可以指定为这些。我会解释它们和根活动的关系。

标准（ `standard` ）

此模式调用的活动不会是根，它属于调用者端的任务。 每次调用时，都会生成活动实例。

`singleTop`

这个启动模式和“标准”相同，除了启动一个活动，它显示在前台任务的最前面时，不会生成实例。

`singleTask`

这个启动模式根据 Affinity 值确定活动所属的任务。 当匹配 `Activity` 的 Affinity 的任务不存在于后台或前台时，新任务随 `Activity` 的实例一起生成。 当任务存在时，它们都不会被生成。 在前者中，已启动的 `Activity` 实例成为根。

`singleInstance`

与 `singleTask` 相同，但以下几点不同。 只有根活动可以属于新生成的任务。 因此，通过此模式启动的活动实例，始终是根活动。 现在，我们需要注意的是，虽然任务已经存在，并且名称和被调用 `Activity` 的 Affinity 相同，但是被调用 `Activity` 的类名和包含在任务中的 `Activity` 的类名是不同的。

从上面我们可以知道，由 `singleTask` 或 `singleInstance` 启动的 `Activity` 有可能成为根。 为了确保应用的安全性，它不应该由这些模式启动。

接下来，我将解释“被调用活动的任务及其启动模式”。 即使 `Activity` 以“标准”模式调用，它也会成为根 `Activity` 。在某些情况下，取决于 `Activity` 所属的任务状态。

例如，考虑被调用 `Activity` 的任务已经在后台运行的情况。 这里的问题是，任务的活动实例以 `singleInstance` 启动，当以“标准”调用的 `Activity` 的 Affinity 与任务相同时，新任务的生成受到现有的 `singleInstance` 活动的限制。但是，当每个活动的类名称相同时，不会生成任务，并使用现有活动实例。在任何情况下，被调用活动都将成为根活动。

如上所述，调用根 `Activity` 的条件很复杂，例如取决于执行状态。 因此，在开发应用时，最好设法以“标准”来调用活动。

这是一个示例，其中发送给私有活动的意图，可以从其他应用中读取。示例代码表明，私有活动的调用方活动以 `singleInstance` 模式启动。 在这个示例代码中，私有活动以“标准”模式启动，但由于调用方 `Activity` 的 `singleInstance` 条件，这个私有活动成为新任务的根 `Activity` 。 此时，发送给私有活动的敏感信息，在任务历史中记录，因此可以从其他应用读取。 仅供参考，调用方活动和私有活动都具有相同的 Affinity。

AndroidManifest.xml（不推荐）

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.activity.singleinstanceactivity"
>

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >

        <!-- Set the launchMode of the root Activity to "singleI
nstance". -->
        <!-- Do not use taskAffinity -->
        <activity
            android:name="org.jssec.android.activity.singleinsta
nceactivity.PrivateUserActivity"
            android:label="@string/app_name"
            android:launchMode="singleInstance"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN
" />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>

        <!-- Private activity -->
        <!-- Set the launchMode to "standard." -->
        <!-- Do not use taskAffinity -->
        <activity
        android:name="org.jssec.android.activity.singleinstancea
ctivity.PrivateActivity"
        android:label="@string/app_name"
        android:exported="false" />
    </application>
</manifest>
```

私有活动仅仅将结果返回个收到的意图。

PrivateActivity.java

```
package org.jssec.android.activity.singleinstanceactivity;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class PrivateActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.private_activity);
        // Handle intent securely, even though the intent sent f
rom the same application.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        String param = getIntent().getStringExtra("PARAM");
        Toast.makeText(this, String.format("Received param: ¥"%s¥
"", param), Toast.LENGTH_LONG).show();
    }

    public void onReturnResultClick(View view) {
        Intent intent = new Intent();
        intent.putExtra("RESULT", "Sensitive Info");
        setResult(RESULT_OK, intent);
        finish();
    }
}
```

在私有活动的调用方，私有活动以“标准”模式启动，意图不带有任何标志。

PrivateUserActivity.java

```
package org.jssec.android.activity.singleinstanceactivity;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class PrivateUserActivity extends Activity {

    private static final int REQUEST_CODE = 1;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.user_activity);
    }

    public void onUseActivityClick(View view) {
        // Start the Private Activity with "standard" lanchMode.
        Intent intent = new Intent(this, PrivateActivity.class);
        intent.putExtra("PARAM", "Sensitive Info");
        startActivityForResult(intent, REQUEST_CODE);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode
, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode != RESULT_OK) return;
        switch (requestCode) {
            case REQUEST_CODE:
                String result = data.getStringExtra("RESULT");
                // Handle received result data carefully and sec
urely,
                // even though the data came from the Activity i
n the same application.
                // Omitted, since this is a sample. Please refer
 to "3.2 Handling Input Data Carefully and Securely."
                Toast.makeText(this, Str();
                break;
        }
    }
}
```

4.1.3.5 使用活动时的日志输出

当使用一个活动时，意图的内容通过 `ActivityManager` 输出到 `LogCat` 。以下内容将被输出到 `LogCat` ，因此在这种情况下，敏感信息不应该包含在这里。

- 目标包名称
- 目标类名称
- 由 `Intent#setData()` 设置的 URI

例如，当应用发送邮件时，如果应用将邮件地址指定为 URI，则邮件地址不幸会输出到 `LogCat` 。所以，最好通过设置 `Extras` 来发送。如下所示发送邮件时，邮件地址会显示给 `logCat` 。

MainActivity.java

```
// URI is output to the LogCat.
Uri uri = Uri.parse("mailtoest@gmail.com");
Intent intent = new Intent(Intent.ACTION_SENDTO, uri);
startActivity(intent);
```

当使用 `Extras` 时，邮件地址不会再展示给 `LogCat` 了。

MainActivity.java

```
// Contents which was set to Extra, is not output to the LogCat.
Uri uri = Uri.parse("mailto:");
Intent intent = new Intent(Intent.ACTION_SENDTO, uri);
intent.putExtra(Intent.EXTRA_EMAIL, new String[] {"test@gmail.co
m"});
startActivity(intent);
```

但是，有些情况下，其他应用可以使用 `ActivityManager#getRecentTasks()` 读取意图的附加数据。请参阅“4.1.2.2 不指定 `taskAffinity` （必需）”，“4.1.2.3 不指定 `launchMode` （必需）”和“4.1.2.4 不要为启动活动的 `Intent` 设置 `FLAG_ACTIVITY_NEW_TASK` 标志（必需）”。

4.1.3.6 防止 `PreferenceActivity` 中的 `Fragment` 注入

当从 `PreferenceActivity` 派生的类是公共活动时，可能会出现称为片段注入 [5] 的问题。为了防止出现这个问题，有必要重写 `PreferenceActivity` .IsValidFragment() ，并检查其参数的有效性，来确保 Activity 不会无意中处理任何 Fragment`。（输入数据安全的更多信息，请参见第3.2节“小心和安全地处理输入数据”。）

> [5] `Fragement` 注入的更多信息，请参考：https://securityintelligence.com/new-vulnerability-android-framework-fragment-injection/ 。

下面我们显示一个覆盖 `IsValidFragment()` 的示例。请注意，如果源代码已被混淆，则类名称和参数值比较的结果可能会更改。在这种情况下，有必要寻求替代对策。

覆盖的 `isValidFragment()` 方法的示例

```
protected boolean isValidFragment(String fragmentName) {
    // If the source code is obfuscated, we must pursue alternat
ive strategies
    return PreferenceFragmentA.class.getName().equals(fragmentNa
me)
        || PreferenceFragmentB.class.getName().equals(fragmentNa
me)
        || PreferenceFragmentC.class.getName().equals(fragmentNa
me)
        || PreferenceFragmentD.class.getName().equals(fragmentNa
me);
}
```

请注意，如果应用的 `targetSdkVersion` 为 19 或更大，不覆盖 `PreferenceActivity.isValidFragment()` 将导致安全异常，并在插入 `Fragment` 时终止应用[调用 `isValidFragment()` 时]，因此在这种情况下，覆盖 `PreferenceActivity.isValidFragment()` 是强制性的。

4.2 接收/发送广播

4.2.1 示例代码

接收广播需要创建广播接收器。 使用广播接收器的风险和对策，根据收到的广播的类型而有所不同。 你可以在以下判断流程中找到你的广播接收器。 接收应用无法检查发送广播的应用的包名称，它是链接伙伴所需的。 因此，无法创建用于伙伴的广播接收器。

表 4.2：广播接收器的类型定义：

类型	定义
私有	只能接收来自相同应用的广播的广播接收器，所以是最安全的
公共	可以接收来自未指定的大量应用的广播的广播接收器
内部	只能接收来自其他内部应用的广播的广播接收器

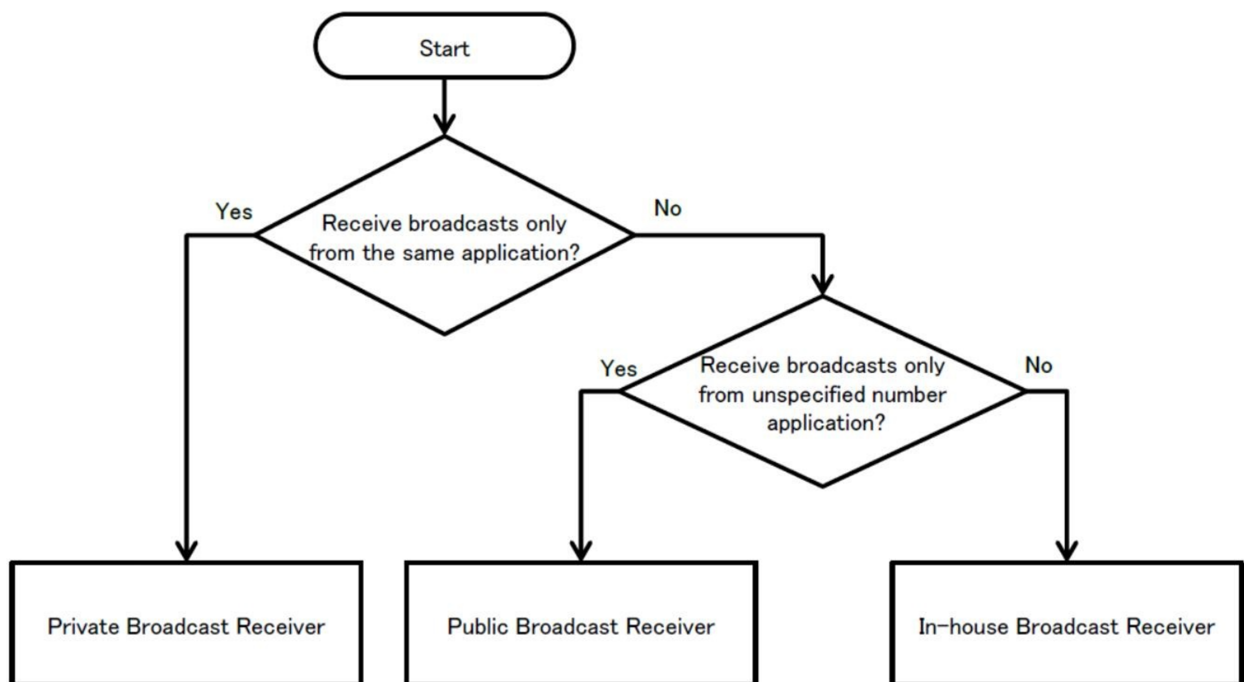


Figure 4.2-1

另外，根据定义方法，广播接收器可以分为两类：静态和动态。 它们之间的差异可以在下图中找到。 示例代码展示了每类的实现方法。 还描述了发送应用的实现方法，因为发送信息的对策取决于接收器来确定。

表 4.2-2

	定义方法	
静态	由 `AndroidManifest.xml` 中的 `<receiver>` 元素定义	1）存在一些 由系统发送的 如 `ACTION_BA` 2）从应用最 前，可以收到
动态	通过在程序中调用 `registerReceiver()` 和 `unregisterReceiver()` ，动态注册和注销广播接收器	1）可以收到静 到的广播。2） 以由程序控制 在前台时，可 能创建私有广

4.2.1.1 私有广播接收器

私人广播接收器是最安全的广播接收器，因为只能接收到从应用内发送的广播。 动态广播接收器不能注册为私有，所以私有广播接收器只包含静态广播接收器。

要点（接收广播）：

1) 将导出属性显示设为 `false`

2) 小心并安全地处理收到的意图，即使意图从相同的应用中发送

3) 敏感信息可以作为返回结果发送，因为请求来自相同应用

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.broadcast.privatereceiver" >

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:allowBackup="false" >
        <!-- Private Broadcast Receiver -->
        <!-- *** POINT 1 *** Explicitly set the exported attribu
te to false. -->
        <receiver
        android:name=".PrivateReceiver"
        android:exported="false" />

        <activity
            android:name=".PrivateSenderActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

PrivateReceiver.java

```
package org.jssec.android.broadcast.privatereceiver;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class PrivateReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        // *** POINT 2 *** Handle the received intent carefully
and securely,
        // even though the intent was sent from within the same
application.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        String param = intent.getStringExtra("PARAM");
        Toast.makeText(context,
        String.format("Received param: ¥"%s¥"", param),
        Toast.LENGTH_SHORT).show();
        // *** POINT 3 *** Sensitive information can be sent as
the returned results since the requests come from within the sam
e application.
        setResultCode(Activity.RESULT_OK);
        setResultData("Sensitive Info from Receiver");
        abortBroadcast();
    }
}
```

向私有广播接收器发送广播的代码展示在下面：

要点（发送广播）：

4) 使用带有指定类的显式意图，来调用相同应用中的接收器。

5) 敏感信息可以发送，因为目标接收器在相同应用中。

6) 小心并安全地处理收到的返回结果，即使数据来自相同应用中的接收器。

PrivateSenderActivity.java

```
package org.jssec.android.broadcast.privatereceiver;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
```

```
import android.widget.TextView;

public class PrivateSenderActivity extends Activity {

    public void onSendNormalClick(View view) {
        // *** POINT 4 *** Use the explicit Intent with class sp
ecified to call a receiver within the same application.
        Intent intent = new Intent(this, PrivateReceiver.class);
        // *** POINT 5 *** Sensitive information can be sent sin
ce the destination Receiver is within the same application.
        intent.putExtra("PARAM", "Sensitive Info from Sender");
        sendBroadcast(intent);
    }

    public void onSendOrderedClick(View view) {
        // *** POINT 4 *** Use the explicit Intent with class sp
ecified to call a receiver within the same application.
        Intent intent = new Intent(this, PrivateReceiver.class);
        // *** POINT 5 *** Sensitive information can be sent sin
ce the destination Receiver is within the same application.
        intent.putExtra("PARAM", "Sensitive Info from Sender");
        sendOrderedBroadcast(intent, null, mResultReceiver, null
, 0, null, null);
    }

    private BroadcastReceiver mResultReceiver = new BroadcastRec
eiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            // *** POINT 6 *** Handle the received result data c
arefully and securely,
            // even though the data came from the Receiver withi
n the same application.
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
            String data = getResultData();
            PrivateSenderActivity.this.logLine(
                String.format("Received result: ¥"%s¥"", data));
        }
    };

    private TextView mLogView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mLogView = (TextView)findViewById(R.id.logview);
    }

    private void logLine(String line) {
        mLogView.append(line);
        mLogView.append("¥n");
    }
```

```
}
```

4.2.1.2 公共广播接收器

公共广播接收器是可以从未指定的大量应用程序接收广播的广播接收器，因此有必要注意，它可能从恶意软件接收广播。

要点（接收广播）：

1) 将导出属性显式设为 `true` 。

2) 小心并安全地处理收到的意图。

3) 返回结果时，不要包含敏感信息。

公共广播接收器的示例代码可以用于静态和动态广播接收器。

PublicReceiver.java

```
package org.jssec.android.broadcast.publicreceiver;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class PublicReceiver extends BroadcastReceiver {

    private static final String MY_BROADCAST_PUBLIC =
    "org.jssec.android.broadcast.MY_BROADCAST_PUBLIC";
    public boolean isDynamic = false;

    private String getName() {
        return isDynamic ? "Public Dynamic Broadcast Receiver" :
"Public Static Broadcast Receiver";
    }

    @Override
    public void onReceive(Context context, Intent intent) {
        // *** POINT 2 *** Handle the received Intent carefully
and securely.
        // Since this is a public broadcast receiver, the reques
ting application may be malware.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        if (MY_BROADCAST_PUBLIC.equals(intent.getAction())) {
            String param = intent.getStringExtra("PARAM");
            Toast.makeText(context,
            String.format("%s:¥nReceived param: ¥"%s¥"", getName
(), param),
            Toast.LENGTH_SHORT).show();
        }
        // *** POINT 3 *** When returning a result, do not inclu
de sensitive information.
        // Since this is a public broadcast receiver, the reques
ting application may be malware.
        // If no problem when the information is taken by malwar
e, it can be returned as result.
        setResultCode(Activity.RESULT_OK);
        setResultData(String.format("Not Sensitive Info from %s"
, getName()));
        abortBroadcast();
    }
}
```

静态广播接收器定义在 `AndroidManifest.xml` 中：

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
package="org.jssec.android.broadcast.publicreceiver" >
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:allowBackup="false" >

        <!-- Public Static Broadcast Receiver -->
        <!-- *** POINT 1 *** Explicitly set the exported attribu
te to true. -->
        <receiver
            android:name=".PublicReceiver"
            android:exported="true" >
            <intent-filter>
            <action android:name="org.jssec.android.broadcast.MY
_BROADCAST_PUBLIC" />
            </intent-filter>
        </receiver>

        <service
            android:name=".DynamicReceiverService"
            android:exported="false" />

        <activity
            android:name=".PublicReceiverActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

在动态广播接收器中，通过调用程序中的 `registerReceiver()` 或 `unregisterReceiver()` 来执行注册/注销。为了通过按钮操作执行注册/注销，该按钮 `PublicReceiverActivity` 中定义。由于动态广播接收器实例的作用域比 `PublicReceiverActivity` 长，因此不能将其保存为 `PublicReceiverActivity` 的成员变量。在这种情况下，请将动态广播接收器实例保存为 `DynamicReceiverService` 的成员变量，然后从 `PublicReceiverActivity` 启动/结束 `DynamicReceiverService` ，来间接注册/注销动态广播接收器。

DynamicReceiverService.java

```
package org.jssec.android.broadcast.publicreceiver;

import android.app.Service;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.IBinder;
import android.widget.Toast;

public class DynamicReceiverService extends Service {

    private static final String MY_BROADCAST_PUBLIC =
    "org.jssec.android.broadcast.MY_BROADCAST_PUBLIC";
    private PublicReceiver mReceiver;

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        // Register Public Dynamic Broadcast Receiver.
        mReceiver = new PublicReceiver();
        mReceiver.isDynamic = true;
        IntentFilter filter = new IntentFilter();
        filter.addAction(MY_BROADCAST_PUBLIC);
        filter.setPriority(1); // Prioritize Dynamic Broadcast R
eceiver, rather than Static Broadcast Receiver.
        registerReceiver(mReceiver, filter);
        Toast.makeText(this,
            "Registered Dynamic Broadcast Receiver.",
        Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        // Unregister Public Dynamic Broadcast Receiver.
        unregisterReceiver(mReceiver);
        mReceiver = null;
        Toast.makeText(this,
        "Unregistered Dynamic Broadcast Receiver.",
        Toast.LENGTH_SHORT).show();
    }
}
```

PublicReceiverActivity.java

```
package org.jssec.android.broadcast.publicreceiver;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class PublicReceiverActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void onRegisterReceiverClick(View view) {
        Intent intent = new Intent(this, DynamicReceiverService.
class);
        startService(intent);
    }

    public void onUnregisterReceiverClick(View view) {
        Intent intent = new Intent(this, DynamicReceiverService.
class);
        stopService(intent);
    }
}
```

接下来，展示了将广播发送到公共广播接收器的示例代码。 当向公共广播接收器发送广播时，需要注意广播可以被恶意软件接收。

要点（发送广播）：

4) 不要发送敏感信息

5) 接受广播时，小心并安全地处理结果数据

PublicSenderActivity.java

```
package org.jssec.android.broadcast.publicsender;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class PublicSenderActivity extends Activity {
```

```
    private static final String MY_BROADCAST_PUBLIC =
    "org.jssec.android.broadcast.MY_BROADCAST_PUBLIC";

    public void onSendNormalClick(View view) {
        // *** POINT 4 *** Do not send sensitive information.
        Intent intent = new Intent(MY_BROADCAST_PUBLIC);
        intent.putExtra("PARAM", "Not Sensitive Info from Sender"
);
        sendBroadcast(intent);
    }

    public void onSendOrderedClick(View view) {
        // *** POINT 4 *** Do not send sensitive information.
        Intent intent = new Intent(MY_BROADCAST_PUBLIC);
        intent.putExtra("PARAM", "Not Sensitive Info from Sender"
);
        sendOrderedBroadcast(intent, null, mResultReceiver, null
, 0, null, null);
    }

    public void onSendStickyClick(View view) {
        // *** POINT 4 *** Do not send sensitive information.
        Intent intent = new Intent(MY_BROADCAST_PUBLIC);
        intent.putExtra("PARAM", "Not Sensitive Info from Sender"
);
        //sendStickyBroadcast is deprecated at API Level 21
        sendStickyBroadcast(intent);
    }

    public void onSendStickyOrderedClick(View view) {
        // *** POINT 4 *** Do not send sensitive information.
        Intent intent = new Intent(MY_BROADCAST_PUBLIC);
        intent.putExtra("PARAM", "Not Sensitive Info from Sender"
);
        //sendStickyOrderedBroadcast is deprecated at API Level
21
        sendStickyOrderedBroadcast(intent, mResultReceiver, null
, 0, null, null);
    }

    public void onRemoveStickyClick(View view) {
        Intent intent = new Intent(MY_BROADCAST_PUBLIC);
        //removeStickyBroadcast is deprecated at API Level 21
        removeStickyBroadcast(intent);
    }

    private BroadcastReceiver mResultReceiver = new BroadcastRec
eiver() {

        @Override
        public void onReceive(Context context, Intent intent) {
            // *** POINT 5 *** When receiving a result, handle t
he result data carefully and securely.
```

```
                // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
                String data = getResultData();
                PublicSenderActivity.this.logLine(
                    String.format("Received result: ¥"%s¥"", data));
            }
        };

    private TextView mLogView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mLogView = (TextView)findViewById(R.id.logview);
    }

    private void logLine(String line) {
        mLogView.append(line);
        mLogView.append("¥n");
    }
}
```

4.2.1.3 内部广播接收器

内部广播接收器是广播接收器，它将永远不会收到从内部应用以外发送的任何广播。 它由几个内部应用组成，用于保护内部应用处理的信息或功能。

要点（接收广播）：

1) 定义内部签名权限来接收广播。

2) 声明使用内部签名权限来接收结果。

3) 将导出属性显式设置为`true`。

4) 需要静态广播接收器定义的内部签名权限。

5) 需要内部签名来注册动态广播接收器。

6) 确认内部签名权限是由内部应用定义的。

7) 尽管广播是从内部应用发送的，但要小心并安全地处理接收到的意图。

8) 由于请求应用是内部的，因此可以返回敏感信息。

9) 导出 APK 时，使用与发送应用相同的开发人员密钥对 APK 进行签名。

内部广播接收器的示例代码可用于静态和动态广播接收器。

InhouseReceiver.java

```
package org.jssec.android.broadcast.inhousereceiver;

import org.jssec.android.shared.SigPerm;
import org.jssec.android.shared.Utils;
import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class InhouseReceiver extends BroadcastReceiver {

    // In-house Signature Permission
    private static final String MY_PERMISSION = "org.jssec.andro
id.broadcast.inhousereceiver.MY_PERMISSION";
    // In-house certificate hash value
    private static String sMyCertHash = null;

    private static String myCertHash(Context context) {
        if (sMyCertHash == null) {
            if (Utils.isDebuggable(context)) {
                // Certificate hash value of "androiddebugkey" i
n the debug.keystore.
```

```
                sMyCertHash = "0EFB7236 328348A9 89718BAD DF57F5
44 D5CCB4AE B9DB34BC 1E29DD26 F77C8255";
            } else {
                // Certificate hash value of "my company key" in
 the keystore.
                sMyCertHash = "D397D343 A5CBC10F 4EDDEB7C A10062
DE 5690984F 1FB9E88B D7B3A7C2 42E142CA";
            }
        }
        return sMyCertHash;
    }

    private static final String MY_BROADCAST_INHOUSE =
    "org.jssec.android.broadcast.MY_BROADCAST_INHOUSE";
    public boolean isDynamic = false;

    private String getName() {
        return isDynamic ? "In-house Dynamic Broadcast Receiver"
 : "In-house Static Broadcast Receiver";
    }

    @Override
    public void onReceive(Context context, Intent intent) {
        // *** POINT 6 *** Verify that the in-house signature pe
rmission is defined by an in-house application.
        if (!SigPerm.test(context, MY_PERMISSION, myCertHash(con
text))) {
            Toast.makeText(context, "The in-house signature perm
ission is not declared by in-house application.",
            Toast.LENGTH_LONG).show();
            return;
        }
        // *** POINT 7 *** Handle the received intent carefully
and securely,
        // even though the Broadcast was sent from an in-house a
pplication..
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        if (MY_BROADCAST_INHOUSE.equals(intent.getAction())) {
            String param = intent.getStringExtra("PARAM");
            Toast.makeText(context,
            String.format("%s:¥nReceived param: ¥"%s¥"", getName
(), param),
            Toast.LENGTH_SHORT).show();
        }
        // *** POINT 8 *** Sensitive information can be returned
 since the requesting application is inhouse.
        setResultCode(Activity.RESULT_OK);
        setResultData(String.format("Sensitive Info from %s", ge
tName()));
        abortBroadcast();
    }
}
```

静态广播接收器定义在 `AndroidManifest.xml` 中。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.broadcast.inhousereceiver" >

    <!-- *** POINT 1 *** Define an in-house signature permission
 to receive Broadcasts -->
    <permission
    android:name="org.jssec.android.broadcast.inhousereceiver.MY
_PERMISSION"
    android:protectionLevel="signature" />
    <!-- *** POINT 2 *** Declare to use the in-house signature p
ermission to receive results. -->
    <uses-permission
    android:name="org.jssec.android.broadcast.inhousesender.MY_P
ERMISSION" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:allowBackup="false" >

        <!-- *** POINT 3 *** Explicitly set the exported attribu
te to true. -->
        <!-- *** POINT 4 *** Require the in-house signature perm
ission by the Static Broadcast Receiver
        definition. -->
        <receiver
            android:name=".InhouseReceiver"
            android:permission="org.jssec.android.broadcast.inho
usereceiver.MY_PERMISSION"
            android:exported="true">
            <intent-filter>
                <action android:name="org.jssec.android.broadcas
t.MY_BROADCAST_INHOUSE" />
                </intent-filter>
        </receiver>

        <service
        android:name=".DynamicReceiverService"
        android:exported="false" />

        <activity
            android:name=".InhouseReceiverActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
```

```
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

在动态广播接收器中，通过调用程序中的 `registerReceiver()` 或 `unregisterReceiver()` 来执行注册/注销。为了通过按钮操作执行注册/注销，该按钮 `PublicReceiverActivity` 中定义。由于动态广播接收器实例的作用域比 `PublicReceiverActivity` 长，因此不能将其保存为 `PublicReceiverActivity` 的成员变量。在这种情况下，请将动态广播接收器实例保存为 `DynamicReceiverService` 的成员变量，然后从 `PublicReceiverActivity` 启动/结束 `DynamicReceiverService` ，来间接注册/注销动态广播接收器。

InhouseReceiverActivity.java

```
package org.jssec.android.broadcast.inhousereceiver;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class InhouseReceiverActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void onRegisterReceiverClick(View view) {
        Intent intent = new Intent(this, DynamicReceiverService.
class);
        startService(intent);
    }

    public void onUnregisterReceiverClick(View view) {
        Intent intent = new Intent(this, DynamicReceiverService.
class);
        stopService(intent);
    }
}
```

DynamicReceiverService.java

```
package org.jssec.android.broadcast.inhousereceiver;

import android.app.Service;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.IBinder;
import android.widget.Toast;

public class DynamicReceiverService extends Service {

    private static final String MY_BROADCAST_INHOUSE =
    "org.jssec.android.broadcast.MY_BROADCAST_INHOUSE";
    private InhouseReceiver mReceiver;

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        mReceiver = new InhouseReceiver();
        mReceiver.isDynamic = true;
        IntentFilter filter = new IntentFilter();
        filter.addAction(MY_BROADCAST_INHOUSE);
        filter.setPriority(1); // Prioritize Dynamic Broadcast R
eceiver, rather than Static Broadcast Receiver.
        // *** POINT 5 *** When registering a dynamic broadcast
receiver, require the in-house signature permission.
        registerReceiver(mReceiver, filter, "org.jssec.android.b
roadcast.inhousereceiver.MY_PERMISSION", null);
        Toast.makeText(this,
            "Registered Dynamic Broadcast Receiver.",
            Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onDestroy() {
        super.onDestroy();

        unregisterReceiver(mReceiver);
        mReceiver = null;
        Toast.makeText(this,
            "Unregistered Dynamic Broadcast Receiver.",
        Toast.LENGTH_SHORT).show();
    }
}
```

SigPerm.java

```
package org.jssec.android.shared;

import android.content.Context;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.PermissionInfo;

public class SigPerm {

    public static boolean test(Context ctx, String sigPermName, String correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, sigPermName));
    }

    public static String hash(Context ctx, String sigPermName) {
        if (sigPermName == null) return null;
        try {
            // Get the package name of the application which declares a permission named sigPermName.
            PackageManager pm = ctx.getPackageManager();
            PermissionInfo pi;
            pi = pm.getPermissionInfo(sigPermName, PackageManager.GET_META_DATA);
            String pkgname = pi.packageName;
            // Fail if the permission named sigPermName is not a Signature Permission
            if (pi.protectionLevel != PermissionInfo.PROTECTION_SIGNATURE) return null;
            // Return the certificate hash value of the application which declares a permission named sigPermName.
            return PkgCert.hash(ctx, pkgname);
        } catch (NameNotFoundException e) {
            return null;
        }
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;
```

```
public class PkgCert {

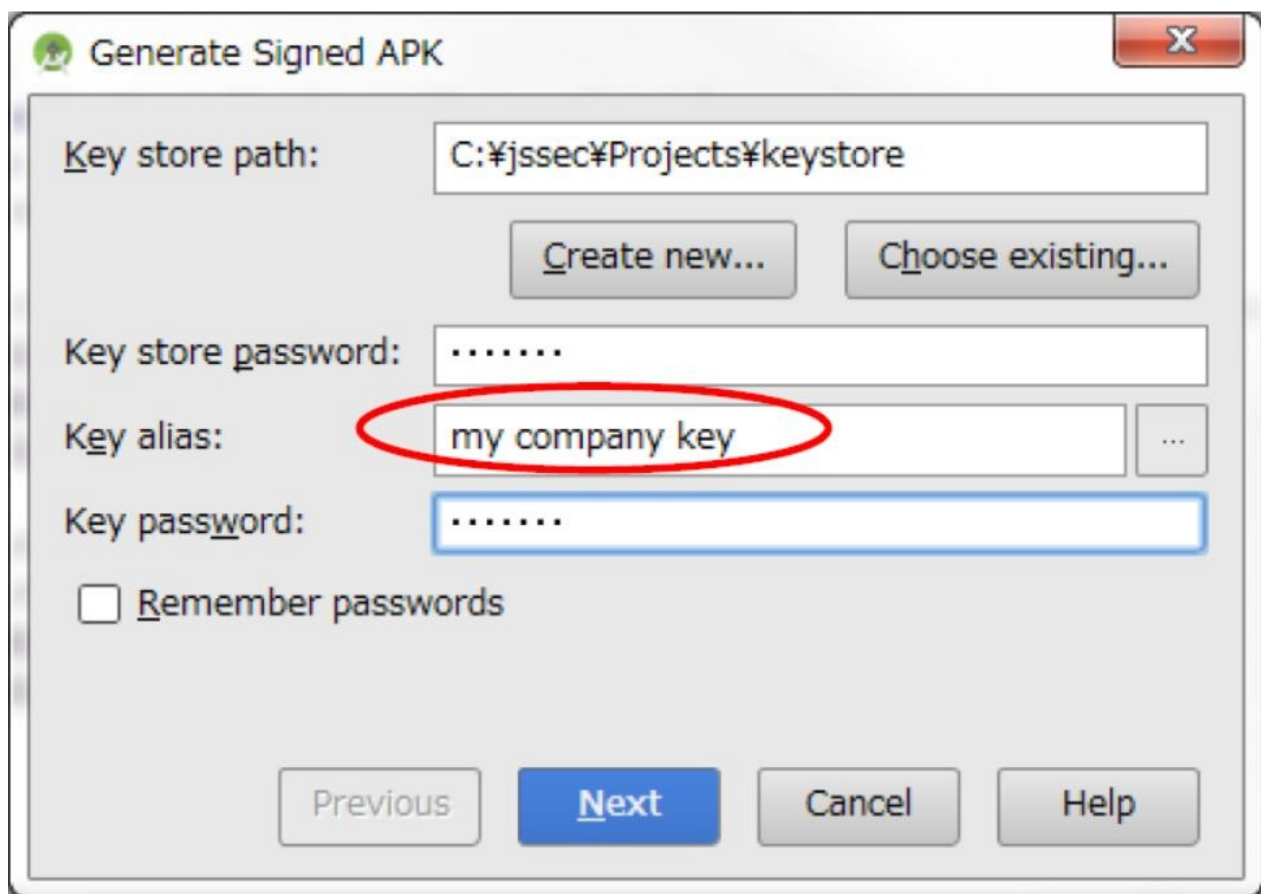
    public static boolean test(Context ctx, String pkgname, Stri
ng correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
    }
}
```

导出 APK 时，使用与发送应用相同的开发人员密钥对 APK 进行签名。



下面，展示了用于向内部广播接收器发送广播的示例代码。

要点（发送广播）：

10) 定义内部签名权限来接收结果。

11) 声明使用内部签名权限来接收广播。

12) 确认内部签名权限是由内部应用定义的。

13) 由于请求应用是内部应用，因此可以返回敏感信息。

14) 需要接收器的内部签名权限。

15) 小心并安全地处理收到的结果数据。

16) 导出 APK 时，请使用与目标应用相同的开发人员密钥对 APK 进行签名。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.broadcast.inhousesender" >

    <uses-permission android:name="android.permission.BROADCAST_
STICKY"/>

    <!-- *** POINT 10 *** Define an in-house signature permissio
n to receive results. -->
    <permission
    android:name="org.jssec.android.broadcast.inhousesender.MY_P
ERMISSION"
    android:protectionLevel="signature" />

    <!-- *** POINT 11 *** Declare to use the in-house signature
permission to receive Broadcasts. -->
    <uses-permission
    android:name="org.jssec.android.broadcast.inhousereceiver.MY
_PERMISSION" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:allowBackup="false" >
        <activity
            android:name="org.jssec.android.broadcast.inhousesen
der.InhouseSenderActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

InhouseSenderActivity.java

```
package org.jssec.android.broadcast.inhousesender;

import org.jssec.android.shared.SigPerm;
import org.jssec.android.shared.Utils;
import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
```

```
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class InhouseSenderActivity extends Activity {

    // In-house Signature Permission
    private static final String MY_PERMISSION = "org.jssec.andro
id.broadcast.inhousesender.MY_PERMISSION";
    // In-house certificate hash value
    private static String sMyCertHash = null;

    private static String myCertHash(Context context) {
        if (sMyCertHash == null) {
            if (Utils.isDebuggable(context)) {
                // Certificate hash value of "androiddebugkey" i
n the debug.keystore.
                sMyCertHash = "0EFB7236 328348A9 89718BAD DF57F5
44 D5CCB4AE B9DB34BC 1E29DD26 F77C8255";
            } else {
                // Certificate hash value of "my company key" in
 the keystore.
                sMyCertHash = "D397D343 A5CBC10F 4EDDEB7C A10062
DE 5690984F 1FB9E88B D7B3A7C2 42E142CA";
            }
        }
        return sMyCertHash;
    }

    private static final String MY_BROADCAST_INHOUSE =
    "org.jssec.android.broadcast.MY_BROADCAST_INHOUSE";

    public void onSendNormalClick(View view) {
        // *** POINT 12 *** Verify that the in-house signature p
ermission is defined by an in-house application.
        if (!SigPerm.test(this, MY_PERMISSION, myCertHash(this))
) {
            Toast.makeText(this, "The in-house signature permiss
ion is not declared by in-house application.",
            Toast.LENGTH_LONG).show();
            return;
        }
        // *** POINT 13 *** Sensitive information can be returne
d since the requesting application is in-house.
        Intent intent = new Intent(MY_BROADCAST_INHOUSE);
        intent.putExtra("PARAM", "Sensitive Info from Sender");
        // *** POINT 14 *** Require the in-house signature permi
ssion to limit receivers.
        sendBroadcast(intent, "org.jssec.android.broadcast.inhou
sesender.MY_PERMISSION");
    }
```

```
    public void onSendOrderedClick(View view) {
        // *** POINT 12 *** Verify that the in-house signature p
ermission is defined by an in-house application.
        if (!SigPerm.test(this, MY_PERMISSION, myCertHash(this))
) {
            Toast.makeText(this, "The in-house signature permiss
ion is not declared by in-house application.",
            Toast.LENGTH_LONG).show();
            return;
        }
        // *** POINT 13 *** Sensitive information can be returne
d since the requesting application is in-house.
        Intent intent = new Intent(MY_BROADCAST_INHOUSE);
        intent.putExtra("PARAM", "Sensitive Info from Sender");
        // *** POINT 14 *** Require the in-house signature permi
ssion to limit receivers.
        sendOrderedBroadcast(intent, "org.jssec.android.broadcas
t.inhousesender.MY_PERMISSION",
        mResultReceiver, null, 0, null, null);
    }

    private BroadcastReceiver mResultReceiver = new BroadcastRec
eiver() {

        @Override
        public void onReceive(Context context, Intent intent) {
            // *** POINT 15 *** Handle the received result data
carefully and securely,
            // even though the data came from an in-house applic
ation.
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
            String data = getResultData();
            InhouseSenderActivity.this.logLine(String.format("Re
ceived result: ¥"%s¥"", data));
        }
    };

    private TextView mLogView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mLogView = (TextView)findViewById(R.id.logview);
    }

    private void logLine(String line) {
        mLogView.append(line);
        mLogView.append("¥n");
    }
}
```

SigPerm.java

```
package org.jssec.android.shared;

import android.content.Context;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.PermissionInfo;

public class SigPerm {

    public static boolean test(Context ctx, String sigPermName,
String correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, sigPermName));
    }

    public static String hash(Context ctx, String sigPermName) {
        if (sigPermName == null) return null;
        try {
            // Get the package name of the application which dec
lares a permission named sigPermName.
            PackageManager pm = ctx.getPackageManager();
            PermissionInfo pi;
            pi = pm.getPermissionInfo(sigPermName, PackageManage
r.GET_META_DATA);
            String pkgname = pi.packageName;
            // Fail if the permission named sigPermName is not a
 Signature Permission
            if (pi.protectionLevel != PermissionInfo.PROTECTION_
SIGNATURE) return null;
            // Return the certificate hash value of the applicat
ion which declares a permission named sigPermName.
            return PkgCert.hash(ctx, pkgname);
        } catch (NameNotFoundException e) {
            return null;
        }
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
```

```
import android.content.pm.Signature;

public class PkgCert {

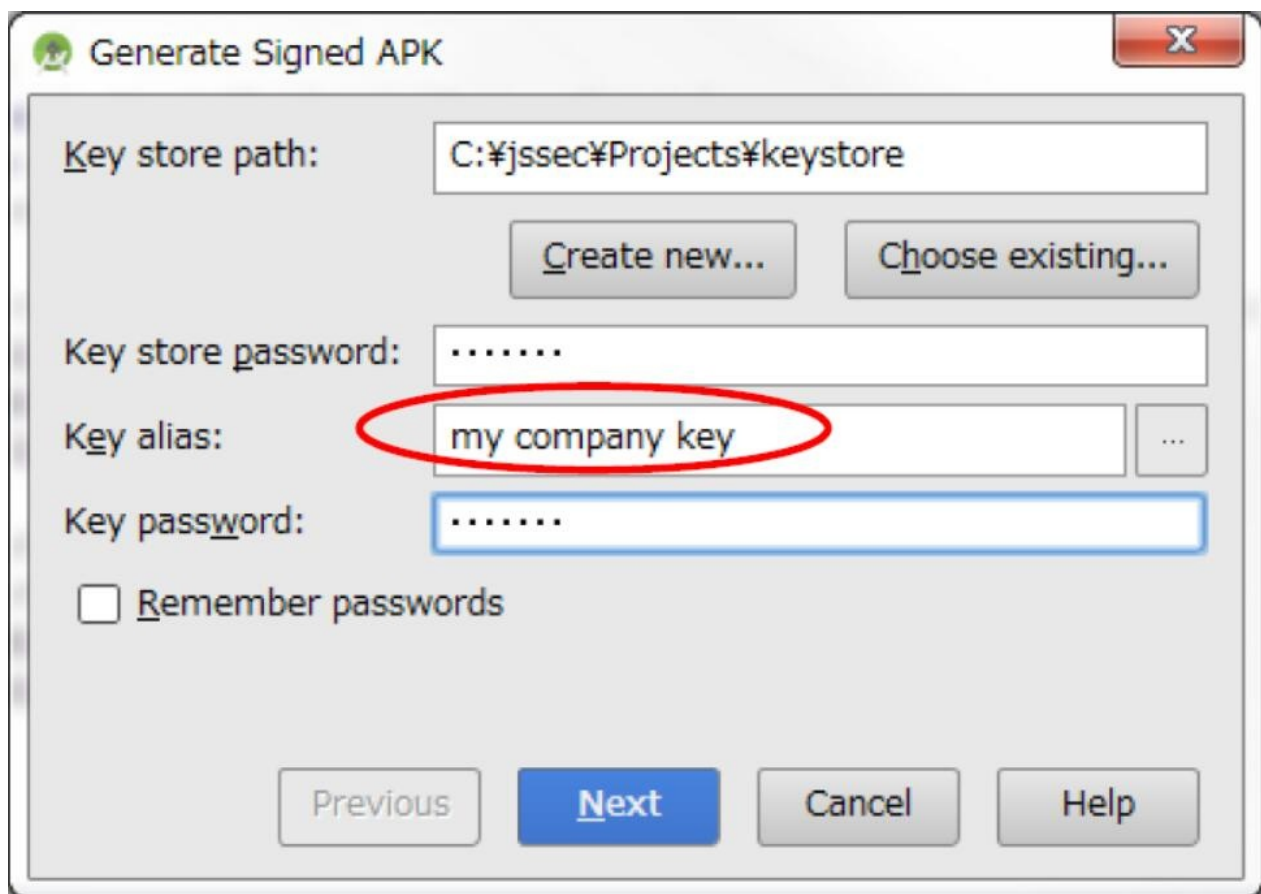
    public static boolean test(Context ctx, String pkgname, Stri
ng correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
    if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
    }
}
```

导出 APK 时，使用与发送应用相同的开发人员密钥对 APK 进行签名。


Generate Signed APK
Key store path:
C:¥jssec¥Projects¥keystore
Create new...
Choose existing...
Key store password:
Key alias:
my company key
Key password:
Remember passwords
Previous
Next
Cancel
Help

4.2.2 规则书

遵循下列规则来发送或接受广播。

4.2.2.1 仅在应用中使用的广播接收器必须设置为私有（必需）

仅在应用中使用的广播接收器应该设置为私有，以避免意外地从其他应用接收任何广播。 它将防止应用功能滥用或异常行为。

仅在同一应用内使用的接收器，不应设计为设置意图过滤器。 由于意图过滤器的特性，即使通过意图过滤器调用同一应用中的私有接收器，其他应用的公共私有也可能被意外调用。

AndroidManifest.xml（不推荐）

```
<!-- Private Broadcast Receiver -->
<!-- *** POINT 1 *** Set the exported attribute to false explici
tly. -->
<receiver
    android:name=".PrivateReceiver"
    android:exported="false" >
    <intent-filter>
        <action android:name="org.jssec.android.broadcast.MY_ACT
ION" />
    </intent-filter>
</receiver>
```

请参阅“4.2.3.1 导出属性和意图过滤器设置的组合（对于接收器）”。

4.2.2.2 小心和安全地处理收到的意图（必需）

虽然风险因广播接收器的类型而异，但处理接收到的意图数据时，首先应该验证意图的安全性。 由于公共广播接收器从未指定的大量应用接收意图，它可能会收到恶意软件的攻击意图。 私有广播接收器将永远不会直接从其他应用接收任何意图，但公共组件从其他应用接收的意图数据，可能会转发到私有广播接收器。 所以不要认为收到的意图在没有任何验证的情况下，是完全安全的。 内部广播接收机具有一定程度的风险，因此还需要验证接收意图的安全性。

请参考“3.2 小心和安全地处理输入数据”。

4.2.2.3 验证签名权限是否由内部应用定义后，使用内部定义的签名权限（必需）

只接收内部应用发送的广播的内部广播接收器，应受内部定义的签名许可保护。 `AndroidManifest.xml` 中的权限定义/权限请求声明不足以保护，因此请参阅“5.2.1.2 如何使用内部定义的签名权限在内部应用之间进行通信”。 通过

对 `receiverPermission` 参数指定内部定义的签名权限来结束广播，需要相同的方式的验证。

4.2.2.4 返回结果信息时，清注意来自目标应用的结果信息泄露（必需）

通过 `setResult()` 返回结果信息的应用的可靠性取决于广播接收器的类型。 对于公共广播接收器，目标应用可能是恶意软件，可能存在恶意使用结果信息的风险。 对于私有广播接收器和内部广播接收器，结果的目的地是内部开发的应用，因此无需介意结果信息的处理。

如上所述，当从广播接收器返回结果信息时，需要注意从目标应用泄漏的结果信息。

4.2.2.5 使用广播发送敏感信息时，限制能收到的接收器（必需）

广播是所创建的系统，用于向未指定的大量应用广播信息或一次通知其时间。 因此，广播敏感信息需要谨慎设计，以防止恶意软件非法获取信息。 对于广播敏感信息，只有可靠的广播接收器可以接收它，而其他广播接收器则不能。 以下是广播发送方法的一些示例。

- 方法是，通过使用显式意图，将广播仅仅发送给预期的可靠广播接收器，来固定地址。
    - 当它发送给同一个应用中的广播接收器时，通过 `Intent#setClass(Context, Class)` 指定地址。 具体代码，请参阅“4.2.1.1 私有广播接收器 - 接收/发送广播”的示例代码部分。
    - 当它发送到其他应用中的广播接收器时，通过 `Intent#setClassName(String, String)` 指定地址。 通过比较目标包中 APK 签名的开发人员密钥和白名单来发送广播，来确认允许的应用。 实际上下面的使用隐式意图的方法更实用。
- 方法是，通过将 `receiverPermission` 指定为内部定义的签名权限，并使可靠的广播接收器声明使用此签名权限，来发送广播。具体代码请参阅“4.2.1.3 内部广播接收器 - 接收/发送广播”的示例代码部分。 另外，实现这种广播发送方法，需要应用规则“4.2.2.3 在验证签名权限由内部应用定义之后，使用内部定义的签名权限”。

4.2.2.6 粘性广播中禁止包含敏感信息（必需）

通常情况下，广播由可用的广播接收器接收后会消失。 另一方面，粘性广播（以下粘性广播包括粘性有序广播）即使由可用的广播接收器接收也不会从系统中消失，并且能够由 `registerReceiver()` 接收。 当粘性广播变得不必要时，可以随时用 `removeStickyBroadcast()` 任意删除它。

由于在预设情况下，粘性广播被隐式意图使用。 具有指定 `receiverPermission` 参数的广播无法发送。 出于这个原因，通过粘性广播发送的信息，可以被多个未指定的应用访问 - 包括恶意软件 - 因此敏感信息禁止以这种方式发送。 请注意，粘性广播在 Android 5.0（API Level 21）中已弃用。

4.2.2.7 注意不指定`receiverPermission`的有序广播无法传递（必需）

不指定`receiverPermission`参数的有序广播，可以由未指定的大量应用接收，包括恶意软件。 有序广播用于接收来自接收器的返回信息，并使几个接收器逐一执行处理。 广播按优先顺序发送给接收器。 因此，如果高优先级恶意软件先接收广播并执行`abortBroadcast()`，则广播将不会传送到后面的接收器。

4.2.2.8 小心并安全地处理来自广播接收器的返回的结果数据（必需）

基本上，考虑到接收结果可能是攻击数据，结果数据应该被安全地处理，尽管风险取决于返回结果数据的广播接收器的类型。

当发送方（源）广播接收器是公共广播接收器时，它从未指定的大量应用接收返回数据。 所以它也可能会收到恶意软件的攻击数据。 当发送方（源）广播接收器是私有广播接收者时，似乎没有风险。 然而，其他应用接收的数据可能会间接作为结果数据转发。 因此，如果没有任何验证，结果数据不应该被认为是安全的。 当发送方（源）广播接收器是内部广播接收器时，它具有一定程度的风险。 因此，考虑到结果数据可能是攻击数据，应该以安全的方式处理它。

请参考“3.2 小心和安全地处理输入数据”。

4.2.2.9 提供二手素材时，素材应该以相同保护级别提供（必需）

当由权限保护的信息或功能素材被二次提供给其他应用时，有必要通过声明与目标应用相同的权限来维持保护标准。 在 Android 权限安全模型中，权限仅管理来自应用的受保护素材的直接访问。 由于这些特点，所得素材可能会被提供给其他应用，而无需声明保护所需的权限。 这实际上与重新授权相同，因为它被称为重新授权问题。 请参阅“5.2.3.4 重新授权问题”。

4.2.3 高级话题

4.2.3.1 结合导出属性和意图过滤器设置（用于接收器）

表 4.2-3 展示了实现接收器时，导出设置和意图过滤器元素的允许的组合。 下面介绍为什么原则上禁止使用带有意图过滤器定义的 `exported ="false"` 。

表 4.2-3 可用与否，导出属性和意图过滤器元素的组合

	导出属性的值		
	True	False	未指定
意图过滤器已定义	OK	不使用	不使用
意图过滤器未定义	OK	OK	不使用

未指定接收器的导出属性时，接收器是否为公共的，取决于该接收器的意图过滤器的存在与否 [6]。但是，在本手册中，禁止将导出的属性设置为不确定的。 通常，如前所述，最好避免依赖任何给定 API 的默认行为的实现；此外，如果存在明确的方法（如导出属性）来启用重要的安全相关设置，那么使用这些方法总是一个好主意。

> [6] 如果意图过滤器已定义，接收器是公共的，否则是私有的。更多信息请参考 https://developer.android.com/guide/topics/manifest/receiver-element.html#exported 。

即使在相同的应用中将广播发送到私有接收器，其他应用中的公共接收器也可能会意外调用。 这就是为什么禁止指定带有意图过滤器定义的 `exported ="false"` 。 以下两张图展示了意外调用的发生情况。

图 4.2-4 是一个正常行为的例子，隐式意图只能在同一个应用中调用私有接收器（应用 A）。 意图过滤器（在图中， `action ="X"` ）仅在应用 A 中定义，所以这是预期的行为。

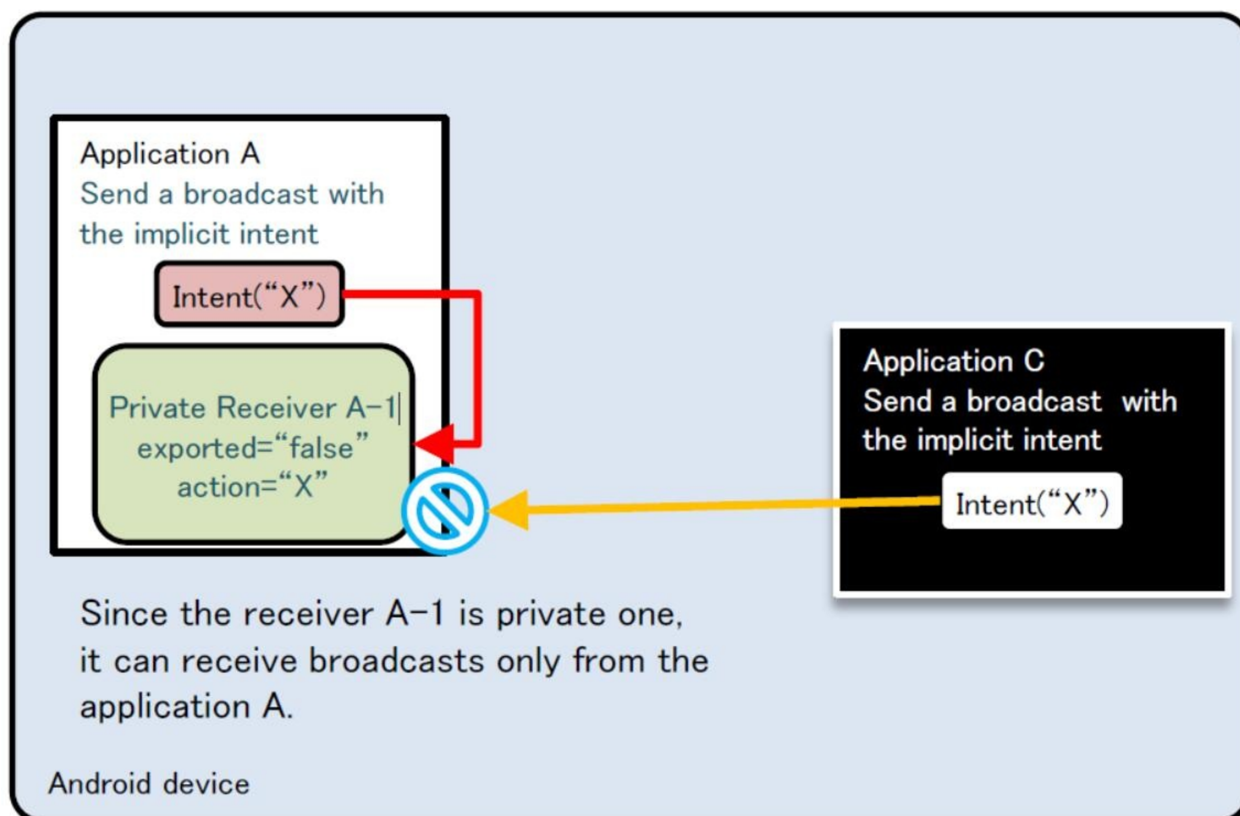


Figure 4.2-4

图 4.2-5 是个例子，应用 B 和应用 A 中都定义了意图过滤器（见图中的 `action ="X"` ）的。首先，当另一个应用（应用 C）通过 隐式意图发送广播，它们不被私有接收器（A-1）接收。 所以不会有任何安全问题。 （请参阅图中的橙色箭头标记。）从安全角度来看，问题是应用 A 对同一应用中的私有接收器的调用。 当应用 A 广播隐式意图时，不仅是相同应用中的私有接收器，而且具有相同意图过滤器定义的公共接收器（B-1）也可以接收意图。 （图中的红色箭头标记）。 在这种情况下，敏感信息可能会从应用 A 发送到 B。当应用 B 是恶意软件时，会导致敏感信息的泄漏。 当发送有序广播时，它可能会收到意外的结果信息。

然而，当广播接收器仅接收由系统发送的广播意图时，应使用带有意图过滤器定义的 `exported="false"` 。 其他组合不应使用。 这是基于这样一个事实，即系统发送的广播意图可以通过 `exported="false"` 来接收。 如果其他应用发送的意图的 `ACTION` 与系统发送的广播意图相同，则可能会通过接收它而导致意外行为。 但是，这可以通过指定 `exported="false"` 来防止。

4.2.3.2 接收器在启动应用之前不会被注册

请务必注意，在 `AndroidManifest.xml` 中定义的静态广播接收器，在安装后不会自动启用 [7]。应用只有在第一次启动后才能接收广播；因此，安装后无法使用接收的广播作为启动操作的触发器。 但是，如果在发送广播时设置了 `Intent.FLAG_INCLUDE_STOPPED_PACKAGES` 标志，则即使是尚未第一次启动的应用也会收到该广播。

> [7] 在 3.0 之前的版本中，接收器可以通过安装 App 自动启动。

4.2.3.3 私有广播接收器可以接收由相同 UID 发送的广播

应用

相同的 UID 可以提供给几个应用。 即使它是私有广播接收器，也可以接收从 UID 相同的应用发送的广播。 但是，这不会是一个安全问题。 由于可以确保 UID 相同的应用具有用于签署 APK 的一致的开发人员密钥。 这意味着私有广播接收器收到的广播，只是从内部应用发送的广播。

4.2.3.4 广播的类型和特性

根据是否有序以及是否粘滞的组合，广播有四种类型。 要发送的广播类型基于广播发送方法而确定。 请注意，粘性广播在 Android 5.0（API Level 21）中已弃用。

类型	发送方法	是否有序	是否粘性
普通	`sendBroadcast()`	否	否
有序	`sendOrderedBroadcast()`	是	否
粘性	`sendStickyBroadcast()`	否	是
粘性有序	`sendStickyOrderedBroadcast()`	是	是

每个广播类型的特性描述如下：

类型	特性
普通	普通广播发送到可接收的广播接收器时消失。 广播由多个广播接收器同时接收。 这与有序广播有所不同。 广播被允许由特定的广播接收机接收。
有序	有序广播的特点是，可接收的广播接收器依次接收广播。 优先级较高的广播接收器较早收到。 当广播被传送到所有广播接收器或广播接收器调用 `abortBroadcast()` ，广播将消失。 广播被允许由声明了特定权限的广播接收器接收。 另外，广播接收器发送的结果信息，可以由发送者使用有序广播接收。 SMS 接收通知的广播（ `SMS_RECEIVED` ）是有序广播的代表性示例。
粘性	粘性广播不会消失并保留在系统中，然后调用 `registerReceiver()` 的应用可以稍后接收粘性广播。 由于粘性广播与其他广播不同，它不会自动消失。 因此，当不需要粘性广播时，需要显式调用 `removeStickyBroadcast()` 来删除粘滞广播。 此外，带有特定权限的受限的广播接收器无法接收广播。 电池状态变化通知的广播（ `ACTION_BATTERY_CHANGED` ）是粘性广播的代表性示例。
粘性有序	这是具有有序和粘性特征的广播。 与粘性广播相同，它不能仅仅允许带有特定权限的广播接收器接收广播。

从广播特性行为的角度来看，上表反过来排列在下面的表中。

广播的特征行为	普通	有序	粘性	粘性有序
由权限限制的广播接收器可以接收广播	OK	OK	-	-
从广播接收器获得过程结果	-	OK	-	OK
使广播接收器按顺序处理广播	-	OK	-	OK
稍后收到已经发送的广播	-	-	OK	OK

4.2.3.5 广播信息可能输出到 `LogCat`

发送/接收的广播基本上不会输出到 `LogCat` 。然而，缺少权限导致接收/发送方的错误时，将输出错误日志。由广播发送的意图信息包含在错误日志中，因此在发生错误之后，需要注意，发送广播时，意图的信息显示在 `LogCat` 中。

发送方的缺少权限的错误：

```
W/ActivityManager(266): Permission Denial: broadcasting Intent {
 act=org.jssec.android.broadcastreceive
r.creating.action.MY_ACTION } from org.jssec.android.broadcast.s
ending (pid=4685, uid=10058) requires o
rg.jssec.android.permission.MY_PERMISSION due to receiver org.js
sec.android.broadcastreceiver.creating/
org.jssec.android.broadcastreceiver.creating.CreatingType3Receiv
er
```

接收方的缺少权限的错误：

```
W/ActivityManager(275): Permission Denial: receiving Intent { ac
t=org.jssec.android.broadcastreceiver.c
reating.action.MY_ACTION } to org.jssec.android.broadcastreceive
r.creating requires org.jssec.android.p
ermission.MY_PERMISSION due to sender org.jssec.android.broadcas
t.sending (uid 10158)
```

4.2.3.6 在主屏幕放置应用的快捷方式时，需要注意的东西

在下面的内容中，我们讨论了创建快捷方式时的一些需要注意的东西，它们用于从主屏幕启动应用，或者用于创建 URL 快捷方式，例如 Web 浏览器中的书签。作为一个例子，我们考虑如下所示的实现。

在主屏幕放置应用的快捷方式：

```
Intent targetIntent = new Intent(this, TargetActivity.class);

// Intent to request shortcut creation

Intent intent = new Intent("com.android.launcher.action.INSTALL_
SHORTCUT");
// Specify an Intent to be launched when the shortcut is tapped
intent.putExtra(Intent.EXTRA_SHORTCUT_INTENT, targetIntent);
Parcelable icon = Intent.ShortcutIconResource.fromContext(contex
t, iconResource);
intent.putExtra(Intent.EXTRA_SHORTCUT_ICON_RESOURCE, icon);
intent.putExtra(Intent.EXTRA_SHORTCUT_NAME, title);
intent.putExtra("duplicate", false);

// Use Broadcast to send the system our request for shortcut cre
ation
context.sendBroadcast(intent);
```

在由上面的代码片段发送的广播中，接收器是主屏幕应用，并且很难识别包名; 我们必须谨慎记住，这是一个向公共接收器传递的隐式意图。 因此，此片段发送的广播，可以被任何任意应用接收，包括恶意软件；因此，在意图中包含敏感信息可能会造成信息泄漏的风险。 特别重要的是要注意，在创建基于 URL 的快捷方式时，秘密信息可能包含在 URL 本身中。

作为对策，有必要遵循“4.2.1.2 公共广播接收器 - 接收/发送广播”中列出的要点，并确保传输的意图不包含敏感信息。

4.3 创建/使用内容供应器

由于 `ContentResolver` 和 `SQLiteDatabase` 的接口非常相似，所以常常有个误解， `Content Provider` 与 `SQLiteDatabase` 的关系如此密切。 但是，实际上内容供应器只是提供了应用间数据共享的接口，所以需要注意的是它不会影响每种数据保存格式。 为了保存内容供应器中的数据，可以使用 `SQLiteDatabase` ，也可以使用其他保存格式，如 XML 文件格式。 以下示例代码中不包含任何数据保存过程，因此请在需要时添加它。

4.3.1 示例代码

使用内容供应器的风险和对策取决于内容供应器的使用方式。 在本节中，我们根据内容供应器的使用方式，对 5 种类型的内容供应器进行了分类。 您可以通过下面显示的图表，找出您应该创建哪种类型的内容供应器。

表 4.3-1 内容供应器类型定义

类型	定义
私有	不能由其他应用使用用的内容供应器，所以是最安全的
公共	应该由未指定的大量应用使用的内容供应器
伙伴	只能由可信的伙伴公司开发的特定应用使用的内容供应器
内容	只能由其它内部应用使用的内容供应器
临时	基本上是私有内容供应器，但允许特定应用访问特定 URI

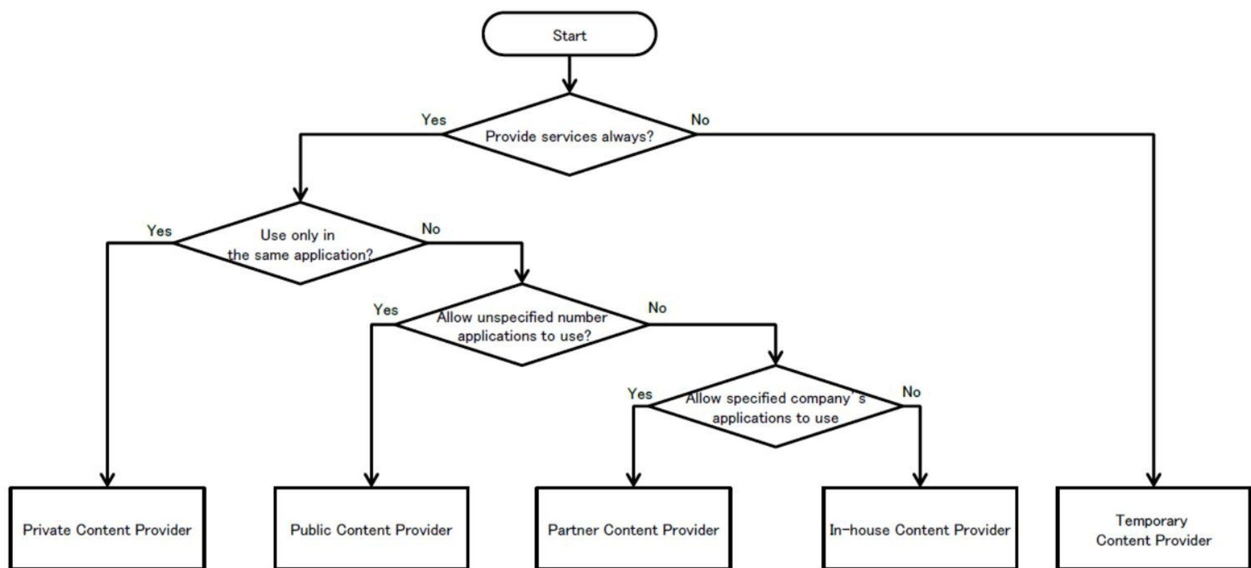


Figure 4.3-1

4.3.1.1 创建/使用私有内容供应器

私有内容供应器是只由单一应用使用的内容提供者，它是最安全的内容供应器 [8]。下面展示了如何实现私有内容供应器的示例代码。

要点（创建内容供应器）：

1) 将导出属性显式设置为 `false`。

2) 即使数据来自相同应用，也应该小心并安全地处理收到的请求数据。

3) 可以发送敏感信息，因为它在同一应用内发送和接收所有信息。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.provider.privateprovider">

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".PrivateUserActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>

        <!-- *** POINT 1 *** Explicitly set the exported attribu
te to false. -->
        <provider
        android:name=".PrivateProvider"
        android:authorities="org.jssec.android.provider.privatep
rovider"
        android:exported="false" />
    </application>
</manifest>
```

PrivateProvider.java

```
package org.jssec.android.provider.privateprovider;
```

```
import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.MatrixCursor;
import android.net.Uri;

public class PrivateProvider extends ContentProvider {

    public static final String AUTHORITY = "org.jssec.android.pr
ovider.privateprovider";
    public static final String CONTENT_TYPE = "vnd.android.curso
r.dir/vnd.org.jssec.contenttype";
    public static final String CONTENT_ITEM_TYPE = "vnd.android.
cursor.item/vnd.org.jssec.contenttype";
    // Expose the interface that the Content Provider provides.

    public interface Download {
        public static final String PATH = "downloads";
        public static final Uri CONTENT_URI = Uri.parse("content
://" + AUTHORITY + "/" + PATH);
    }

    public interface Address {
        public static final String PATH = "addresses";
        public static final Uri CONTENT_URI = Uri.parse("content
://" + AUTHORITY + "/" + PATH);
    }

    // UriMatcher
    private static final int DOWNLOADS_CODE = 1;
    private static final int DOWNLOADS_ID_CODE = 2;
    private static final int ADDRESSES_CODE = 3;
    private static final int ADDRESSES_ID_CODE = 4;
    private static UriMatcher sUriMatcher;

    static {
        sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        sUriMatcher.addURI(AUTHORITY, Download.PATH, DOWNLOADS_C
ODE);
        sUriMatcher.addURI(AUTHORITY, Download.PATH + "/#", DOWN
LOADS_ID_CODE);
        sUriMatcher.addURI(AUTHORITY, Address.PATH, ADDRESSES_CO
DE);
        sUriMatcher.addURI(AUTHORITY, Address.PATH + "/#", ADDRE
SSES_ID_CODE);
    }

    // Since this is a sample program,
    // query method returns the following fixed result always wi
thout using database.
```

```
    private static MatrixCursor sAddressCursor = new MatrixCurso
r(new String[] { "_id", "city" });

    static {
        sAddressCursor.addRow(new String[] { "1", "New York" });
        sAddressCursor.addRow(new String[] { "2", "Longon" });
        sAddressCursor.addRow(new String[] { "3", "Paris" });
    }

    private static MatrixCursor sDownloadCursor = new MatrixCurs
or(new String[] { "_id", "path" });

    static {
        sDownloadCursor.addRow(new String[] { "1", "/sdcard/down
loads/sample.jpg" });
        sDownloadCursor.addRow(new String[] { "2", "/sdcard/down
loads/sample.txt" });
    }

    @Override
    public boolean onCreate() {
        return true;
    }

    @Override
    public String getType(Uri uri) {
        // *** POINT 2 *** Handle the received request data care
fully and securely,
        // even though the data comes from the same application.
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Please refer to "3.2 Handle Input Data Carefully and
Securely."
        // *** POINT 3 *** Sensitive information can be sent sin
ce it is sending and receiving all within the same application.
        // However, the result of getType rarely has the sensiti
ve meaning.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
            case ADDRESSES_CODE:
                return CONTENT_TYPE;
            case DOWNLOADS_ID_CODE:
            case ADDRESSES_ID_CODE:
                return CONTENT_ITEM_TYPE;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
```

```
    public Cursor query(Uri uri, String[] projection, String sel
ection,
        String[] selectionArgs, String sortOrder) {
        // *** POINT 2 *** Handle the received request data care
fully and securely,
        // even though the data comes from the same application.
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Please refer to "3.2 Handle Input Data Carefully and
Securely."
        // *** POINT 3 *** Sensitive information can be sent sin
ce it is sending and receiving all within the same application.
        // It depends on application whether the query result ha
s sensitive meaning or not.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
            case DOWNLOADS_ID_CODE:
                return sDownloadCursor;
            case ADDRESSES_CODE:
            case ADDRESSES_ID_CODE:
                return sAddressCursor;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        // *** POINT 2 *** Handle the received request data care
fully and securely,
        // even though the data comes from the same application.
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Please refer to "3.2 Handle Input Data Carefully and
Securely."
        // *** POINT 3 *** Sensitive information can be sent sin
ce it is sending and receiving all within the same application.
        // It depends on application whether the issued ID has s
ensitive meaning or not.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
                return ContentUris.withAppendedId(Download.CONTE
NT_URI, 3);
            case ADDRESSES_CODE:
                return ContentUris.withAppendedId(Address.CONTEN
T_URI, 4);
            default:
                throw new IllegalArgumentException("Invalid URI:"
```

```
 + uri);
        }
    }

    @Override
    public int update(Uri uri, ContentValues values, String sele
ction,
        String[] selectionArgs) {
        // *** POINT 2 *** Handle the received request data care
fully and securely,
        // even though the data comes from the same application.
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Please refer to "3.2 Handle Input Data Carefully and
Securely."
        // *** POINT 3 *** Sensitive information can be sent sin
ce it is sending and receiving all within the same application.
        // It depends on application whether the number of updat
ed records has sensitive meaning or not.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
                return 5; // Return number of updated records
            case DOWNLOADS_ID_CODE:
                return 1;
            case ADDRESSES_CODE:
                return 15;
            case ADDRESSES_ID_CODE:
                return 1;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public int delete(Uri uri, String selection, String[] select
ionArgs) {
        // *** POINT 2 *** Handle the received request data care
fully and securely,
        // even though the data comes from the same application.
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Please refer to "3.2 Handle Input Data Carefully and
Securely."
        // *** POINT 3 *** Sensitive information can be sent sin
ce it is sending and receiving all within the same application.
        // It depends on application whether the number of delet
ed records has sensitive meaning or not.
        switch (sUriMatcher.match(uri)) {
```

```
                case DOWNLOADS_CODE:
                    return 10; // Return number of deleted records
                case DOWNLOADS_ID_CODE:
                    return 1;
                case ADDRESSES_CODE:
                    return 20;
                case ADDRESSES_ID_CODE:
                    return 1;
                default:
                    throw new IllegalArgumentException("Invalid URI:"
 + uri);
            }
        }

}
```

下面是活动的示例，它使用私有内容供应器。

要点（使用内容供应器）：

4) 敏感信息可以发送，因为目标供应器在相同应用中。

5) 小心和安全地处理收到的结果数据，即使数据来自相同应用。

PrivateUserActivity.java

```
package org.jssec.android.provider.privateprovider;

import android.app.Activity;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class PrivateUserActivity extends Activity {

    public void onQueryClick(View view) {
        logLine("[Query]");
        Cursor cursor = null;
        try {
            // *** POINT 4 *** Sensitive information can be sent
 since the destination provider is in the same application.
            cursor = getContentResolver().query(
                PrivateProvider.Download.CONTENT_URI, null, null
, null, null);
            // *** POINT 5 *** Handle received result data caref
ully and securely,
            // even though the data comes from the same applicat
ion.
            // Omitted, since this is a sample. Please refer to
```

```
"3.2 Handling Input Data Carefully and Securely."
            if (cursor == null) {
                logLine(" null cursor");
            } else {
                boolean moved = cursor.moveToFirst();
                while (moved) {
                    logLine(String.format(" %d, %s", cursor.getI
nt(0), cursor.getString(1)));
                    moved = cursor.moveToNext();
                }
            }
        }
        finally {
            if (cursor != null) cursor.close();
        }
    }

    public void onInsertClick(View view) {
        logLine("[Insert]");
        // *** POINT 4 *** Sensitive information can be sent sin
ce the destination provider is in the same application.
        Uri uri = getContentResolver().insert(PrivateProvider.Do
wnload.CONTENT_URI, null);
        // *** POINT 5 *** Handle received result data carefully
 and securely,
        // even though the data comes from the same application.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        logLine(" uri:" + uri);
    }

    public void onUpdateClick(View view) {
    logLine("[Update]");
        // *** POINT 4 *** Sensitive information can be sent sin
ce the destination provider is in the same application.
        int count = getContentResolver().update(PrivateProvider.
Download.CONTENT_URI, null, null, null);
        // *** POINT 5 *** Handle received result data carefully
 and securely,
        // even though the data comes from the same application.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        logLine(String.format(" %s records updated", count));
    }

    public void onDeleteClick(View view) {
    logLine("[Delete]");
        // *** POINT 4 *** Sensitive information can be sent sin
ce the destination provider is in the same application.
        int count = getContentResolver().delete(
        PrivateProvider.Download.CONTENT_URI, null, null);
        // *** POINT 5 *** Handle received result data carefully
 and securely,
```

```
        // even though the data comes from the same application.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        logLine(String.format(" %s records deleted", count));
    }

    private TextView mLogView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mLogView = (TextView)findViewById(R.id.logview);
    }

    private void logLine(String line) {
        mLogView.append(line);
        mLogView.append("¥n");
    }
}
```

4.3.1.2 创建/使用公共内容供应器

公共内容供应器是应该由未指定的大量应用使用的内容供应器。 需要注意的是，由于它不指定客户端，它可能会受到恶意软件的攻击和篡改。 例如，可以通过 `select()` 获取保存的数据，可以通过 `update()` 更改数据，或者可以通过 `insert()` / `delete()` 插入/删除假数据。

另外，在使用 Android OS 未提供的自定义公共内容供应器时，需要注意的是，恶意软件可能会接收到请求参数，伪装成自定义公共内容供应器，并且也可能发送攻击数据。 Android OS 提供的联系人和 MediaStore 也是公共内容提供商，但恶意软件不能伪装成它们。

实现公共内容供应器的样例代码展示在下面：

要点（创建内容供应器）：

1) 将导出的属性显式设置为 `true` 。

2) 仔细安全地处理收到的请求数据。

3) 返回结果时，请勿包含敏感信息。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.provider.publicprovider">

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >

        <!-- *** POINT 1 *** Explicitly set the exported attribu
te to true. -->
        <provider
        android:name=".PublicProvider"
        android:authorities="org.jssec.android.provider.publicpr
ovider"
        android:exported="true" />
    </application>
</manifest>
```

PublicProvider.java

```
package org.jssec.android.provider.publicprovider;

import android.content.ContentProvider;
import android.content.ContentUris;
```

```
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.MatrixCursor;
import android.net.Uri;

public class PublicProvider extends ContentProvider {

    public static final String AUTHORITY = "org.jssec.android.pr
ovider.publicprovider";
    public static final String CONTENT_TYPE = "vnd.android.curso
r.dir/vnd.org.jssec.contenttype";
    public static final String CONTENT_ITEM_TYPE = "vnd.android.
cursor.item/vnd.org.jssec.contenttype";
    // Expose the interface that the Content Provider provides.

    public interface Download {
        public static final String PATH = "downloads";
        public static final Uri CONTENT_URI = Uri.parse("content
://" + AUTHORITY + "/" + PATH);
    }

    public interface Address {
        public static final String PATH = "addresses";
        public static final Uri CONTENT_URI = Uri.parse("content
://" + AUTHORITY + "/" + PATH);
    }

    // UriMatcher
    private static final int DOWNLOADS_CODE = 1;
    private static final int DOWNLOADS_ID_CODE = 2;
    private static final int ADDRESSES_CODE = 3;
    private static final int ADDRESSES_ID_CODE = 4;
    private static UriMatcher sUriMatcher;

    static {
        sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        sUriMatcher.addURI(AUTHORITY, Download.PATH, DOWNLOADS_C
ODE);
        sUriMatcher.addURI(AUTHORITY, Download.PATH + "/#", DOWN
LOADS_ID_CODE);
        sUriMatcher.addURI(AUTHORITY, Address.PATH, ADDRESSES_CO
DE);
        sUriMatcher.addURI(AUTHORITY, Address.PATH + "/#", ADDRE
SSES_ID_CODE);
    }

    // Since this is a sample program,
    // query method returns the following fixed result always wi
thout using database.
    private static MatrixCursor sAddressCursor = new MatrixCurso
r(new String[] { "_id", "city" });
```

```
    static {
        sAddressCursor.addRow(new String[] { "1", "New York" });
        sAddressCursor.addRow(new String[] { "2", "London" });
        sAddressCursor.addRow(new String[] { "3", "Paris" });
    }

    private static MatrixCursor sDownloadCursor = new MatrixCurs
or(new String[] { "_id", "path" });

    static {
        sDownloadCursor.addRow(new String[] { "1", "/sdcard/down
loads/sample.jpg" });
        sDownloadCursor.addRow(new String[] { "2", "/sdcard/down
loads/sample.txt" });
    }

    @Override
    public boolean onCreate() {
        return true;
    }

    @Override
    public String getType(Uri uri) {
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
            case ADDRESSES_CODE:
                return CONTENT_TYPE;
            case DOWNLOADS_ID_CODE:
            case ADDRESSES_ID_CODE:
                return CONTENT_ITEM_TYPE;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String sel
ection,
    String[] selectionArgs, String sortOrder) {
        // *** POINT 2 *** Handle the received request data care
fully and securely.
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Refer to "3.2 Handle Input Data Carefully and Securel
y."
        // *** POINT 3 *** When returning a result, do not inclu
de sensitive information.
        // It depends on application whether the query result ha
s sensitive meaning or not.
        // If no problem when the information is taken by malwar
```

```
e, it can be returned as result.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
            case DOWNLOADS_ID_CODE:
                return sDownloadCursor;
            case ADDRESSES_CODE:
            case ADDRESSES_ID_CODE:
                return sAddressCursor;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        // *** POINT 2 *** Handle the received request data care
fully and securely.
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Refer to "3.2 Handle Input Data Carefully and Securel
y."
        // *** POINT 3 *** When returning a result, do not inclu
de sensitive information.
        // It depends on application whether the issued ID has s
ensitive meaning or not.
        // If no problem when the information is taken by malwar
e, it can be returned as result.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
                return ContentUris.withAppendedId(Download.CONTE
NT_URI, 3);
            case ADDRESSES_CODE:
                return ContentUris.withAppendedId(Address.CONTEN
T_URI, 4);
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public int update(Uri uri, ContentValues values, String sele
ction,
        String[] selectionArgs) {
        // *** POINT 2 *** Handle the received request data care
fully and securely.
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
```

```
        // Refer to "3.2 Handle Input Data Carefully and Securel
y."
        // *** POINT 3 *** When returning a result, do not inclu
de sensitive information.
        // It depends on application whether the number of updat
ed records has sensitive meaning or not.
        // If no problem when the information is taken by malwar
e, it can be returned as result.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
                return 5; // Return number of updated records
            case DOWNLOADS_ID_CODE:
                return 1;
            case ADDRESSES_CODE:
                return 15;
            case ADDRESSES_ID_CODE:
                return 1;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public int delete(Uri uri, String selection, String[] select
ionArgs) {
        // *** POINT 2 *** Handle the received request data care
fully and securely.
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Refer to "3.2 Handle Input Data Carefully and Securel
y."
        // *** POINT 3 *** When returning a result, do not inclu
de sensitive information.
        // It depends on application whether the number of delet
ed records has sensitive meaning or not.
        // If no problem when the information is taken by malwar
e, it can be returned as result.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
                return 10; // Return number of deleted records
            case DOWNLOADS_ID_CODE:
                return 1;
            case ADDRESSES_CODE:
                return 20;
            case ADDRESSES_ID_CODE:
                return 1;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
```

```
    }
}
```

下面是使用公共内容供应器的活动示例。

要点（使用内容供应器）：

4) 不要发送敏感信息

5) 收到结果时，小心和安全地处理结果数据

PublicUserActivity.java

```
package org.jssec.android.provider.publicuser;

import android.app.Activity;
import android.content.ContentValues;
import android.content.pm.ProviderInfo;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class PublicUserActivity extends Activity {

    // Target Content Provider Information
    private static final String AUTHORITY = "org.jssec.android.p
rovider.publicprovider";

    private interface Address {
        public static final String PATH = "addresses";
        public static final Uri CONTENT_URI = Uri.parse("content
://" + AUTHORITY + "/" + PATH);
    }

    public void onQueryClick(View view) {
        logLine("[Query]");
        if (!providerExists(Address.CONTENT_URI)) {
            logLine(" Content Provider doesn't exist.");
            return;
        }
        Cursor cursor = null;
        try {
            // *** POINT 4 *** Do not send sensitive information.

            // since the target Content Provider may be malware.
            // If no problem when the information is taken by ma
lware, it can be included in the request.
            cursor = getContentResolver().query(Address.CONTENT_
URI, null, null, null, null);
```

```
            // *** POINT 5 *** When receiving a result, handle t
he result data carefully and securely.
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
            if (cursor == null) {
                logLine(" null cursor");
            } else {
                boolean moved = cursor.moveToFirst();
                while (moved) {
                    logLine(String.format(" %d, %s", cursor.getI
nt(0), cursor.getString(1)));
                    moved = cursor.moveToNext();
                }
            }
        }
        finally {
            if (cursor != null) cursor.close();
        }
    }

    public void onInsertClick(View view) {
    logLine("[Insert]");
        if (!providerExists(Address.CONTENT_URI)) {
            logLine(" Content Provider doesn't exist.");
            return;
        }
        // *** POINT 4 *** Do not send sensitive information.
        // since the target Content Provider may be malware.
        // If no problem when the information is taken by malwar
e, it can be included in the request.
        ContentValues values = new ContentValues();
        values.put("city", "Tokyo");
        Uri uri = getContentResolver().insert(Address.CONTENT_UR
I, values);
        // *** POINT 5 *** When receiving a result, handle the r
esult data carefully and securely.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        logLine(" uri:" + uri);
    }

    public void onUpdateClick(View view) {
    logLine("[Update]");
        if (!providerExists(Address.CONTENT_URI)) {
            logLine(" Content Provider doesn't exist.");
            return;
        }
        // *** POINT 4 *** Do not send sensitive information.
        // since the target Content Provider may be malware.
        // If no problem when the information is taken by malwar
e, it can be included in the request.
        ContentValues values = new ContentValues();
        values.put("city", "Tokyo");
```

```
        String where = "_id = ?";
        String[] args = { "4" };
        int count = getContentResolver().update(Address.CONTENT_
URI, values, where, args);
        // *** POINT 5 *** When receiving a result, handle the r
esult data carefully and securely.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        logLine(String.format(" %s records updated", count));
    }

    public void onDeleteClick(View view) {
    logLine("[Delete]");
        if (!providerExists(Address.CONTENT_URI)) {
            logLine(" Content Provider doesn't exist.");
            return;
        }
        // *** POINT 4 *** Do not send sensitive information.
        // since the target Content Provider may be malware.
        // If no problem when the information is taken by malwar
e, it can be included in the request.
        int count = getContentResolver().delete(Address.CONTENT_
URI, null, null);
        // *** POINT 5 *** When receiving a result, handle the r
esult data carefully and securely.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        logLine(String.format(" %s records deleted", count));
    }

    private boolean providerExists(Uri uri) {
        ProviderInfo pi = getPackageManager().resolveContentProv
ider(uri.getAuthority(), 0);
        return (pi != null);
    }

    private TextView mLogView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mLogView = (TextView)findViewById(R.id.logview);
    }

    private void logLine(String line) {
        mLogView.append(line);
        mLogView.append("¥n");
    }
}
```

4.3.1.3 创建/使用伙伴内容供应器

合作伙伴内容供应器只能由特定应用使用。 该系统由伙伴公司的应用和内部应用组成，用于保护在伙伴应用和内部应用之间处理的信息和功能。

下面显示了用于实现伙伴内容供应器的示例代码。

要点（创建内容供应器）：

1) 将导出属性显式设置为 `true` 。

2) 验证请求应用的证书是否已在自己的白名单中注册。

3) 即使数据来自伙伴应用，也要小心并安全地处理收到的请求数据。

4) 可以返回开放给伙伴应用的信息。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.provider.partnerprovider">
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <!-- *** POINT 1 *** Explicitly set the exported attribu
te to true. -->
        <provider
        android:name=".PartnerProvider"
        android:authorities="org.jssec.android.provider.partnerp
rovider"
        android:exported="true" />
    </application>
</manifest>
```

PartnerProvider.java

```
package org.jssec.android.provider.partnerprovider;

import java.util.List;
import org.jssec.android.shared.PkgCertWhitelists;
import org.jssec.android.shared.Utils;
import android.app.ActivityManager;
import android.app.ActivityManager.RunningAppProcessInfo;
import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.UriMatcher;
```

```
import android.database.Cursor;
import android.database.MatrixCursor;
import android.net.Uri;
import android.os.Binder;
import android.os.Build;

public class PartnerProvider extends ContentProvider {

    public static final String AUTHORITY = "org.jssec.android.pr
ovider.partnerprovider";
    public static final String CONTENT_TYPE = "vnd.android.curso
r.dir/vnd.org.jssec.contenttype";
    public static final String CONTENT_ITEM_TYPE = "vnd.android.
cursor.item/vnd.org.jssec.contenttype";
    // Expose the interface that the Content Provider provides.

    public interface Download {
        public static final String PATH = "downloads";
        public static final Uri CONTENT_URI = Uri.parse("content
://" + AUTHORITY + "/" + PATH);
    }

    public interface Address {
    public static final String PATH = "addresses";
    public static final Uri CONTENT_URI = Uri.parse("content://"
 + AUTHORITY + "/" + PATH);
    }
    // UriMatcher
    private static final int DOWNLOADS_CODE = 1;
    private static final int DOWNLOADS_ID_CODE = 2;
    private static final int ADDRESSES_CODE = 3;
    private static final int ADDRESSES_ID_CODE = 4;
    private static UriMatcher sUriMatcher;

    static {
        sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        sUriMatcher.addURI(AUTHORITY, Download.PATH, DOWNLOADS_C
ODE);
        sUriMatcher.addURI(AUTHORITY, Download.PATH + "/#", DOWN
LOADS_ID_CODE);
        sUriMatcher.addURI(AUTHORITY, Address.PATH, ADDRESSES_CO
DE);
        sUriMatcher.addURI(AUTHORITY, Address.PATH + "/#", ADDRE
SSES_ID_CODE);
    }

    // Since this is a sample program,
    // query method returns the following fixed result always wi
thout using database.
    private static MatrixCursor sAddressCursor = new MatrixCurso
r(new String[] { "_id", "city" });

    static {
```

```
        sAddressCursor.addRow(new String[] { "1", "New York" });
        sAddressCursor.addRow(new String[] { "2", "London" });
        sAddressCursor.addRow(new String[] { "3", "Paris" });
    }

    private static MatrixCursor sDownloadCursor = new MatrixCurs
or(new String[] { "_id", "path" });

    static {
        sDownloadCursor.addRow(new String[] { "1", "/sdcard/down
loads/sample.jpg" });
        sDownloadCursor.addRow(new String[] { "2", "/sdcard/down
loads/sample.txt" });
    }

    // *** POINT 2 *** Verify if the certificate of a requesting
 application has been registered in the
    own white list.
    private static PkgCertWhitelists sWhitelists = null;

    private static void buildWhitelists(Context context) {
        boolean isdebug = Utils.isDebuggable(context);
        sWhitelists = new PkgCertWhitelists();
        // Register certificate hash value of partner applicatio
n org.jssec.android.provider.partneruser.
            sWhitelists.add("org.jssec.android.provider.partneru
ser", isdebug ?
            // Certificate hash value of "androiddebugkey" in th
e debug.keystore.
            "0EFB7236 328348A9 89718BAD DF57F544 D5CCB4AE B9DB34
BC 1E29DD26 F77C8255" :
            // Certificate hash value of "partner key" in the ke
ystore.
            "1F039BB5 7861C27A 3916C778 8E78CE00 690B3974 3EB825
9F E2627B8D 4C0EC35A");
            // Register following other partner applications in
the same way.
    }

    private static boolean checkPartner(Context context, String
pkgname) {
        if (sWhitelists == null) buildWhitelists(context);
        return sWhitelists.test(context, pkgname);
    }

    // Get the package name of the calling application.
    private String getCallingPackage(Context context) {
        String pkgname;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT)
 {
            pkgname = super.getCallingPackage();
        } else {
            pkgname = null;
```

```
            ActivityManager am = (ActivityManager) context.getSy
stemService(Context.ACTIVITY_SERVICE);
            List<RunningAppProcessInfo> procList = am.getRunning
AppProcesses();
            int callingPid = Binder.getCallingPid();
            if (procList != null) {
                for (RunningAppProcessInfo proc : procList) {
                    if (proc.pid == callingPid) {
                        pkgname = proc.pkgList[proc.pkgList.leng
th - 1];
                        break;
                    }
                }
            }
        }
        return pkgname;
    }

    @Override
    public boolean onCreate() {
        return true;
    }

    @Override
    public String getType(Uri uri) {
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
            case ADDRESSES_CODE:
                return CONTENT_TYPE;
            case DOWNLOADS_ID_CODE:
            case ADDRESSES_ID_CODE:
                return CONTENT_ITEM_TYPE;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String sel
ection,
        String[] selectionArgs, String sortOrder) {
        // *** POINT 2 *** Verify if the certificate of a reques
ting application has been registered in the own white list.
        if (!checkPartner(getContext(), getCallingPackage(getCon
text()))) {
            throw new SecurityException("Calling application is
not a partner application.");
        }
        // *** POINT 3 *** Handle the received request data care
fully and securely,
        // even though the data comes from a partner application.
```

```
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Refer to "3.2 Handle Input Data Carefully and Securel
y."
        // *** POINT 4 *** Information that is granted to disclo
se to partner applications can be returned.
        // It depends on application whether the query result ca
n be disclosed or not.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
            case DOWNLOADS_ID_CODE:
                return sDownloadCursor;
            case ADDRESSES_CODE:
            case ADDRESSES_ID_CODE:
                return sAddressCursor;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        // *** POINT 2 *** Verify if the certificate of a reques
ting application has been registered in the own white list.
        if (!checkPartner(getContext(), getCallingPackage(getCon
text()))) {
            throw new SecurityException("Calling application is
not a partner application.");
        }
        // *** POINT 3 *** Handle the received request data care
fully and securely,
        // even though the data comes from a partner application.

        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Refer to "3.2 Handle Input Data Carefully and Securel
y."
        // *** POINT 4 *** Information that is granted to disclo
se to partner applications can be returned.
        // It depends on application whether the issued ID has s
ensitive meaning or not.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
                return ContentUris.withAppendedId(Download.CONTE
NT_URI, 3);
            case ADDRESSES_CODE:
                return ContentUris.withAppendedId(Address.CONTEN
T_URI, 4);
```

```
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public int update(Uri uri, ContentValues values, String sele
ction,
        String[] selectionArgs) {
        // *** POINT 2 *** Verify if the certificate of a reques
ting application has been registered in the own white list.
        if (!checkPartner(getContext(), getCallingPackage(getCon
text()))) {
            throw new SecurityException("Calling application is
not a partner application.");
        }
        // *** POINT 3 *** Handle the received request data care
fully and securely,
        // even though the data comes from a partner application.

        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Refer to "3.2 Handle Input Data Carefully and Securel
y."
        // *** POINT 4 *** Information that is granted to disclo
se to partner applications can be returned.
        // It depends on application whether the number of updat
ed records has sensitive meaning or not.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
                return 5; // Return number of updated records
            case DOWNLOADS_ID_CODE:
                return 1;
            case ADDRESSES_CODE:
                return 15;
            case ADDRESSES_ID_CODE:
                return 1;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public int delete(Uri uri, String selection, String[] select
ionArgs) {
        // *** POINT 2 *** Verify if the certificate of a reques
ting application has been registered in the own white list.
        if (!checkPartner(getContext(), getCallingPackage(getCon
text()))) {
```

```
            throw new SecurityException("Calling application is
not a partner application.");
        }
        // *** POINT 3 *** Handle the received request data care
fully and securely,
        // even though the data comes from a partner application.

        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Refer to "3.2 Handle Input Data Carefully and Securel
y."
        // *** POINT 4 *** Information that is granted to disclo
se to partner applications can be returned.
        // It depends on application whether the number of delet
ed records has sensitive meaning or not.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
                return 10; // Return number of deleted records
            case DOWNLOADS_ID_CODE:
                return 1;
            case ADDRESSES_CODE:
                return 20;
            case ADDRESSES_ID_CODE:
                return 1;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }
}
```

下面是使用伙伴内容供应器的活动示例：

要点（使用内容供应器）：

5) 验证目标应用的证书是否已在自己的白名单中注册。

6) 可以发送开放给伙伴应用的信息。

7) 即使数据来自伙伴应用，也要小心并安全地处理收到的结果数据。

PartnerActivity.java

```
package org.jssec.android.provider.partneruser;

import org.jssec.android.shared.PkgCertWhitelists;
import org.jssec.android.shared.Utils;
import android.app.Activity;
import android.content.ContentValues;
```

```
import android.content.Context;
import android.content.pm.ProviderInfo;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class PartnerUserActivity extends Activity {

    // Target Content Provider Information
    private static final String AUTHORITY = "org.jssec.android.p
rovider.partnerprovider";

    private interface Address {
        public static final String PATH = "addresses";
        public static final Uri CONTENT_URI = Uri.parse("content
://" + AUTHORITY + "/" + PATH);
    }

    // *** POINT 4 *** Verify if the certificate of the target a
pplication has been registered in the own white list.
    private static PkgCertWhitelists sWhitelists = null;

    private static void buildWhitelists(Context context) {
        boolean isdebug = Utils.isDebuggable(context);
        sWhitelists = new PkgCertWhitelists();
        // Register certificate hash value of partner applicatio
n org.jssec.android.provider.partnerprovider.
        sWhitelists.add("org.jssec.android.provider.partnerprovi
der", isdebug ?
            // Certificate hash value of "androiddebugkey" in th
e debug.keystore.
            "0EFB7236 328348A9 89718BAD DF57F544 D5CCB4AE B9DB34
BC 1E29DD26 F77C8255" :
            // Certificate hash value of "partner key" in the ke
ystore.
            "D397D343 A5CBC10F 4EDDEB7C A10062DE 5690984F 1FB9E8
8B D7B3A7C2 42E142CA");
            // Register following other partner applications in
the same way.
    }

    private static boolean checkPartner(Context context, String
pkgname) {
        if (sWhitelists == null) buildWhitelists(context);
        return sWhitelists.test(context, pkgname);
    }

    // Get package name of target content provider.
    private String providerPkgname(Uri uri) {
        String pkgname = null;
        ProviderInfo pi = getPackageManager().resolveContentProv
```

```
ider(uri.getAuthority(), 0);
        if (pi != null) pkgname = pi.packageName;
        return pkgname;
    }

    public void onQueryClick(View view) {
        logLine("[Query]");
        // *** POINT 4 *** Verify if the certificate of the targ
et application has been registered in the own white list.
        if (!checkPartner(this, providerPkgname(Address.CONTENT_
URI))) {
            logLine(" The target content provider is not served
by partner applications.");
            return;
        }
        Cursor cursor = null;
        try {
            // *** POINT 5 *** Information that is granted to di
sclose to partner applications can be sent.
            cursor = getContentResolver().query(Address.CONTENT_
URI, null, null, null, null);
            // *** POINT 6 *** Handle the received result data c
arefully and securely,
            // even though the data comes from a partner applica
tion.
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
            if (cursor == null) {
                logLine(" null cursor");
            } else {
                boolean moved = cursor.moveToFirst();
                while (moved) {
                    logLine(String.format(" %d, %s", cursor.getI
nt(0), cursor.getString(1)));
                    moved = cursor.moveToNext();
                }
            }
        }
        finally {
            if (cursor != null) cursor.close();
        }
    }

    public void onInsertClick(View view) {
        logLine("[Insert]");
        // *** POINT 4 *** Verify if the certificate of the targ
et application has been registered in the own white list.
        if (!checkPartner(this, providerPkgname(Address.CONTENT_
URI))) {
            logLine(" The target content provider is not served
by partner applications.");
            return;
        }
```

```
        // *** POINT 5 *** Information that is granted to disclo
se to partner applications can be sent.
        ContentValues values = new ContentValues();
        values.put("city", "Tokyo");
        Uri uri = getContentResolver().insert(Address.CONTENT_UR
I, values);
        // *** POINT 6 *** Handle the received result data caref
ully and securely,
        // even though the data comes from a partner application.

        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        logLine(" uri:" + uri);
    }

    public void onUpdateClick(View view) {
        logLine("[Update]");
        // *** POINT 4 *** Verify if the certificate of the targ
et application has been registered in the own white list.
        if (!checkPartner(this, providerPkgname(Address.CONTENT_
URI))) {
            logLine(" The target content provider is not served
by partner applications.");
            return;
        }
        // *** POINT 5 *** Information that is granted to disclo
se to partner applications can be sent.
        ContentValues values = new ContentValues();
        values.put("city", "Tokyo");
        String where = "_id = ?";
        String[] args = { "4" };
        int count = getContentResolver().update(Address.CONTENT_
URI, values, where, args);
        // *** POINT 6 *** Handle the received result data caref
ully and securely,
        // even though the data comes from a partner application.

        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        logLine(String.format(" %s records updated", count));
    }

    public void onDeleteClick(View view) {
        logLine("[Delete]");
        // *** POINT 4 *** Verify if the certificate of the targ
et application has been registered in the own white list.
        if (!checkPartner(this, providerPkgname(Address.CONTENT_
URI))) {
            logLine(" The target content provider is not served
by partner applications.");
            return;
        }
        // *** POINT 5 *** Information that is granted to disclo
```

```
se to partner applications can be sent.
        int count = getContentResolver().delete(Address.CONTENT_
URI, null, null);
        // *** POINT 6 *** Handle the received result data caref
ully and securely,
        // even though the data comes from a partner application.

        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        logLine(String.format(" %s records deleted", count));
    }

    private TextView mLogView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mLogView = (TextView)findViewById(R.id.logview);
    }

    private void logLine(String line) {
        mLogView.append(line);
        mLogView.append("¥n");
    }
}
```

PkgCertWhitelists.java

```
package org.jssec.android.shared;

import java.util.HashMap;
import java.util.Map;
import android.content.Context;

public class PkgCertWhitelists {

    private Map<String, String> mWhitelists = new HashMap<String
, String>();

    public boolean add(String pkgname, String sha256) {
        if (pkgname == null) return false;
        if (sha256 == null) return false;
        sha256 = sha256.replaceAll(" ", "");
        if (sha256.length() != 64) return false; // SHA-256 -> 3
2 bytes -> 64 chars
        sha256 = sha256.toUpperCase();
        if (sha256.replaceAll("[0-9A-F]+", "").length() != 0) re
turn false; // found non hex char
        mWhitelists.put(pkgname, sha256);
        return true;
    }

    public boolean test(Context ctx, String pkgname) {
        // Get the correct hash value which corresponds to pkgna
me.
        String correctHash = mWhitelists.get(pkgname);
        // Compare the actual hash value of pkgname with the cor
rect hash value.
        return PkgCert.test(ctx, pkgname, correctHash);
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;

public class PkgCert {

    public static boolean test(Context ctx, String pkgname, Stri
ng correctHash) {
        if (correctHash == null) return false;
```

```
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
    }
}
```

4.3.1.4 创建/使用内部内容供应器

内部内容供应器禁止除内部应用以外的应用使用。

下面展示了如何实现内部内容供应器的示例代码。

要点（创建内容供应器）：

1) 定义内部签名权限。

2) 需要内部签名权限。

3) 将导出属性显式设置为 `true`。

4) 验证内部签名权限是否由内部应用定义。

5) 验证参数的安全性，即使这是来自内部应用的请求。

6) 由于请求应用是内部的，因此可以返回敏感信息。

7) 导出 APK 时，请使用与请求应用相同的开发者密钥对 APK 进行签名。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.provider.inhouseprovider">
    <!-- *** POINT 1 *** Define an in-house signature permission
 -->
    <permission
    android:name="org.jssec.android.provider.inhouseprovider.MY_
PERMISSION"
    android:protectionLevel="signature" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <!-- *** POINT 2 *** Require the in-house signature perm
ission -->
        <!-- *** POINT 3 *** Explicitly set the exported attribu
te to true. -->
        <provider
        android:name=".InhouseProvider"
        android:authorities="org.jssec.android.provider.inhousep
rovider"
        android:permission="org.jssec.android.provider.inhousepr
ovider.MY_PERMISSION"
        android:exported="true" />
    </application>
</manifest>
```

InhouseProvider.java

```
package org.jssec.android.provider.inhouseprovider;

import org.jssec.android.shared.SigPerm;
import org.jssec.android.shared.Utils;
import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.MatrixCursor;
import android.net.Uri;

public class InhouseProvider extends ContentProvider {

    public static final String AUTHORITY = "org.jssec.android.pr
ovider.inhouseprovider";
    public static final String CONTENT_TYPE = "vnd.android.curso
r.dir/vnd.org.jssec.contenttype";
    public static final String CONTENT_ITEM_TYPE = "vnd.android.
cursor.item/vnd.org.jssec.contenttype";
    // Expose the interface that the Content Provider provides.

    public interface Download {
        public static final String PATH = "downloads";
        public static final Uri CONTENT_URI = Uri.parse("content
://" + AUTHORITY + "/" + PATH);
    }

    public interface Address {
        public static final String PATH = "addresses";
        public static final Uri CONTENT_URI = Uri.parse("content
://" + AUTHORITY + "/" + PATH);
    }

    // UriMatcher
    private static final int DOWNLOADS_CODE = 1;
    private static final int DOWNLOADS_ID_CODE = 2;
    private static final int ADDRESSES_CODE = 3;
    private static final int ADDRESSES_ID_CODE = 4;
    private static UriMatcher sUriMatcher;

    static {
        sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        sUriMatcher.addURI(AUTHORITY, Download.PATH, DOWNLOADS_C
ODE);
        sUriMatcher.addURI(AUTHORITY, Download.PATH + "/#", DOWN
LOADS_ID_CODE);
        sUriMatcher.addURI(AUTHORITY, Address.PATH, ADDRESSES_CO
DE);
```

```
        sUriMatcher.addURI(AUTHORITY, Address.PATH + "/#", ADDRE
SSES_ID_CODE);
    }

    // Since this is a sample program,
    // query method returns the following fixed result always wi
thout using database.
    private static MatrixCursor sAddressCursor = new MatrixCurso
r(new String[] { "_id", "city" });

    static {
        sAddressCursor.addRow(new String[] { "1", "New York" });
        sAddressCursor.addRow(new String[] { "2", "London" });
        sAddressCursor.addRow(new String[] { "3", "Paris" });
    }

    private static MatrixCursor sDownloadCursor = new MatrixCurs
or(new String[] { "_id", "path" });

    static {
        sDownloadCursor.addRow(new String[] { "1", "/sdcard/down
loads/sample.jpg" });
        sDownloadCursor.addRow(new String[] { "2", "/sdcard/down
loads/sample.txt" });
    }

    // In-house Signature Permission
    private static final String MY_PERMISSION = "org.jssec.andro
id.provider.inhouseprovider.MY_PERMISSION";
    // In-house certificate hash value
    private static String sMyCertHash = null;

    private static String myCertHash(Context context) {
        if (sMyCertHash == null) {
            if (Utils.isDebuggable(context)) {
                // Certificate hash value of "androiddebugkey" i
n the debug.keystore.
                sMyCertHash = "0EFB7236 328348A9 89718BAD DF57F5
44 D5CCB4AE B9DB34BC 1E29DD26 F77C8255";
            } else {
                // Certificate hash value of "my company key" in
 the keystore.
                sMyCertHash = "D397D343 A5CBC10F 4EDDEB7C A10062
DE 5690984F 1FB9E88B D7B3A7C2 42E142CA";
            }
        }
        return sMyCertHash;
    }

    @Override
    public boolean onCreate() {
        return true;
    }
```

```
    @Override
    public String getType(Uri uri) {
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
            case ADDRESSES_CODE:
                return CONTENT_TYPE;
            case DOWNLOADS_ID_CODE:
            case ADDRESSES_ID_CODE:
                return CONTENT_ITEM_TYPE;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String sel
ection,
        String[] selectionArgs, String sortOrder) {
        // *** POINT 4 *** Verify if the in-house signature perm
ission is defined by an in-house application.
        if (!SigPerm.test(getContext(), MY_PERMISSION, myCertHas
h(getContext()))) {
            throw new SecurityException("The in-house signature
permission is not declared by in-house application.");
        }
        // *** POINT 5 *** Handle the received request data care
fully and securely,
        // even though the data came from an in-house applicatio
n.
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Refer to "3.2 Handle Input Data Carefully and Securel
y."
        // *** POINT 6 *** Sensitive information can be returned
 since the requesting application is inhouse.
        // It depends on application whether the query result ha
s sensitive meaning or not.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
            case DOWNLOADS_ID_CODE:
                return sDownloadCursor;
            case ADDRESSES_CODE:
            case ADDRESSES_ID_CODE:
                return sAddressCursor;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }
```

```
    @Override
    public Uri insert(Uri uri, ContentValues values) {
        // *** POINT 4 *** Verify if the in-house signature perm
ission is defined by an in-house application.
        if (!SigPerm.test(getContext(), MY_PERMISSION, myCertHas
h(getContext()))) {
            throw new SecurityException("The in-house signature
permission is not declared by in-house application.");
        }
        // *** POINT 5 *** Handle the received request data care
fully and securely,
        // even though the data came from an in-house applicatio
n.
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Refer to "3.2 Handle Input Data Carefully and Securel
y."
        // *** POINT 6 *** Sensitive information can be returned
 since the requesting application is inhouse.
        // It depends on application whether the issued ID has s
ensitive meaning or not.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
                return ContentUris.withAppendedId(Download.CONTE
NT_URI, 3);
            case ADDRESSES_CODE:
                return ContentUris.withAppendedId(Address.CONTEN
T_URI, 4);
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public int update(Uri uri, ContentValues values, String sele
ction,
        String[] selectionArgs) {
        // *** POINT 4 *** Verify if the in-house signature perm
ission is defined by an in-house application.
        if (!SigPerm.test(getContext(), MY_PERMISSION, myCertHas
h(getContext()))) {
            throw new SecurityException("The in-house signature
permission is not declared by in-house application.");
        }
        // *** POINT 5 *** Handle the received request data care
fully and securely,
        // even though the data came from an in-house applicatio
n.
        // Here, whether uri is within expectations or not, is v
```

```
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Refer to "3.2 Handle Input Data Carefully and Securel
y."
        // *** POINT 6 *** Sensitive information can be returned
 since the requesting application is inhouse.
        // It depends on application whether the number of updat
ed records has sensitive meaning or not.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
                return 5; // Return number of updated records
            case DOWNLOADS_ID_CODE:
                return 1;
            case ADDRESSES_CODE:
                return 15;
            case ADDRESSES_ID_CODE:
                return 1;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public int delete(Uri uri, String selection, String[] select
ionArgs) {
        // *** POINT 4 *** Verify if the in-house signature perm
ission is defined by an in-house application.
        if (!SigPerm.test(getContext(), MY_PERMISSION, myCertHas
h(getContext()))) {
            throw new SecurityException("The in-house signature
permission is not declared by in-house application.");
        }
        // *** POINT 5 *** Handle the received request data care
fully and securely,
        // even though the data came from an in-house applicatio
n.
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Refer to "3.2 Handle Input Data Carefully and Securel
y."
        // *** POINT 6 *** Sensitive information can be returned
 since the requesting application is inhouse.
        // It depends on application whether the number of delet
ed records has sensitive meaning or not.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
                return 10; // Return number of deleted records
            case DOWNLOADS_ID_CODE:
                return 1;
```

```
            case ADDRESSES_CODE:
                return 20;
            case ADDRESSES_ID_CODE:
                return 1;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }
}
```

SigPerm.java

```
package org.jssec.android.shared;

import android.content.Context;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.PermissionInfo;

public class SigPerm {

    public static boolean test(Context ctx, String sigPermName,
String correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, sigPermName));
    }

    public static String hash(Context ctx, String sigPermName) {
        if (sigPermName == null) return null;
        try {
            // Get the package name of the application which dec
lares a permission named sigPermName.
            PackageManager pm = ctx.getPackageManager();
            PermissionInfo pi;
            pi = pm.getPermissionInfo(sigPermName, PackageManage
r.GET_META_DATA);
            String pkgname = pi.packageName;
            // Fail if the permission named sigPermName is not a
 Signature Permission
            if (pi.protectionLevel != PermissionInfo.PROTECTION_
SIGNATURE) return null;
            // Return the certificate hash value of the applicat
ion which declares a permission named sigPermName.
            return PkgCert.hash(ctx, pkgname);
        } catch (NameNotFoundException e) {
            return null;
        }
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;
```

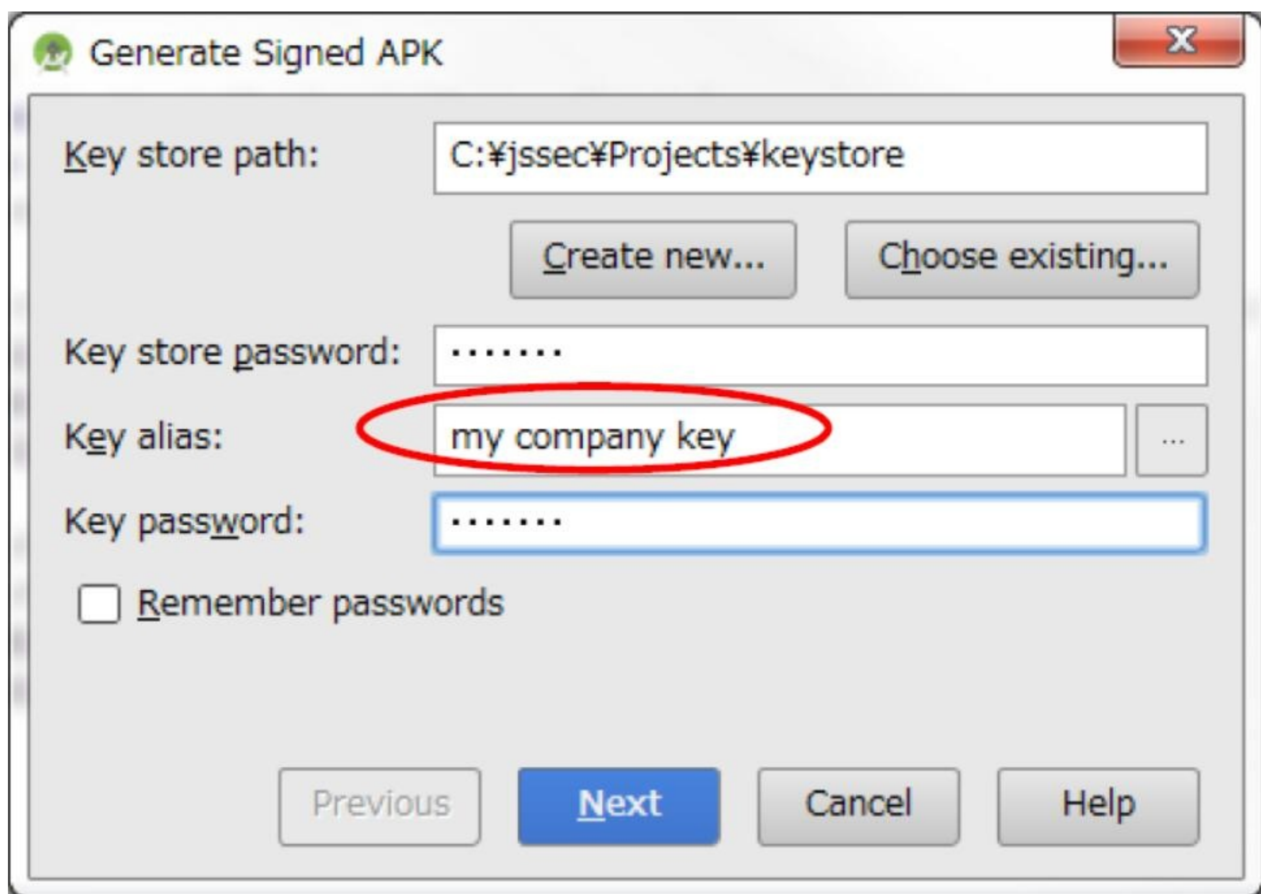
```
public class PkgCert {

    public static boolean test(Context ctx, String pkgname, Stri
ng correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }
    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
    }
}
```

要点 7：导出 APK 时，请使用与请求应用相同的开发者密钥对 APK 进行签名。



下面是使用内部内容供应器的活动示例。

要点（使用内容个供应器）：

8) 声明使用内部签名权限。

9) 验证内部签名权限是否由内部应用定义。

10) 验证目标应用是否使用内部证书签名。

11) 由于目标应用是内部应用，因此可以发送敏感信息。

12) 即使数据来自内部应用，也要小心并安全地处理收到的结果数据。

13) 导出 APK 时，请使用与目标应用相同的开发人员密钥对 APK 进行签名。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.provider.inhouseuser">
    <!-- *** POINT 8 *** Declare to use the in-house signature p
ermission. -->
    <uses-permission
    android:name="org.jssec.android.provider.inhouseprovider.MY_
PERMISSION" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".InhouseUserActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

InhouseUserActivity.java

```
package org.jssec.android.provider.inhouseuser;

import org.jssec.android.shared.PkgCert;
import org.jssec.android.shared.SigPerm;
import org.jssec.android.shared.Utils;
import android.app.Activity;
import android.content.ContentValues;
import android.content.Context;
import android.content.pm.PackageManager;
import android.content.pm.ProviderInfo;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class InhouseUserActivity extends Activity {

    // Target Content Provider Information
    private static final String AUTHORITY = "org.jssec.android.p
rovider.inhouseprovider";
```

```
    private interface Address {
        public static final String PATH = "addresses";
        public static final Uri CONTENT_URI = Uri.parse("content
://" + AUTHORITY + "/" + PATH);
    }

    // In-house Signature Permission
    private static final String MY_PERMISSION = "org.jssec.andro
id.provider.inhouseprovider.MY_PERMISSION";
    // In-house certificate hash value
    private static String sMyCertHash = null;

    private static String myCertHash(Context context) {
        if (sMyCertHash == null) {
            if (Utils.isDebuggable(context)) {
                // Certificate hash value of "androiddebugkey" i
n the debug.keystore.
                sMyCertHash = "0EFB7236 328348A9 89718BAD DF57F5
44 D5CCB4AE B9DB34BC 1E29DD26 F77C8255";
            } else {
                // Certificate hash value of "my company key" in
 the keystore.
                sMyCertHash = "D397D343 A5CBC10F 4EDDEB7C A10062
DE 5690984F 1FB9E88B D7B3A7C2 42E142CA";
            }
        }
        return sMyCertHash;
    }

    // Get package name of target content provider.
    private static String providerPkgname(Context context, Uri u
ri) {
        String pkgname = null;
        PackageManager pm = context.getPackageManager();
        ProviderInfo pi = pm.resolveContentProvider(uri.getAutho
rity(), 0);
        if (pi != null) pkgname = pi.packageName;
        return pkgname;
    }

    public void onQueryClick(View view) {
        logLine("[Query]");
        // *** POINT 9 *** Verify if the in-house signature perm
ission is defined by an in-house application.
        if (!SigPerm.test(this, MY_PERMISSION, myCertHash(this))
) {
            logLine(" The in-house signature permission is not d
eclared by in-house application.");
            return;
        }
        // *** POINT 10 *** Verify if the destination applicatio
n is signed with the in-house certificate.
        String pkgname = providerPkgname(this, Address.CONTENT_U
```

```
RI);
        if (!PkgCert.test(this, pkgname, myCertHash(this))) {
            logLine(" The target content provider is not served
by in-house applications.");
            return;
        }
        Cursor cursor = null;
        try {
            // *** POINT 11 *** Sensitive information can be sen
t since the destination application is in-house one.
            cursor = getContentResolver().query(Address.CONTENT_
URI, null, null, null, null);
            // *** POINT 12 *** Handle the received result data
carefully and securely,
            // even though the data comes from an in-house appli
cation.
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
                if (cursor == null) {
            logLine(" null cursor");
            } else {
                boolean moved = cursor.moveToFirst();
                while (moved) {
                    logLine(String.format(" %d, %s", cursor.getI
nt(0), cursor.getString(1)));
                    moved = cursor.moveToNext();
                }
            }
        } finally {
            if (cursor != null) cursor.close();
        }
    }

    public void onInsertClick(View view) {
        logLine("[Insert]");
        // *** POINT 9 *** Verify if the in-house signature perm
ission is defined by an in-house application.
        String correctHash = myCertHash(this);
        if (!SigPerm.test(this, MY_PERMISSION, correctHash)) {
            logLine(" The in-house signature permission is not d
eclared by in-house application.");
            return;
        }
        // *** POINT 10 *** Verify if the destination applicatio
n is signed with the in-house certificate.
        String pkgname = providerPkgname(this, Address.CONTENT_U
RI);
        if (!PkgCert.test(this, pkgname, correctHash)) {
            logLine(" The target content provider is not served
by in-house applications.");
            return;
        }
        // *** POINT 11 *** Sensitive information can be sent si
```

```
nce the destination application is in-house one.
        ContentValues values = new ContentValues();
        values.put("city", "Tokyo");
        Uri uri = getContentResolver().insert(Address.CONTENT_UR
I, values);
        // *** POINT 12 *** Handle the received result data care
fully and securely,
        // even though the data comes from an in-house applicati
on.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        logLine(" uri:" + uri);
    }

    public void onUpdateClick(View view) {
        logLine("[Update]");
        // *** POINT 9 *** Verify if the in-house signature perm
ission is defined by an in-house application.
        String correctHash = myCertHash(this);
        if (!SigPerm.test(this, MY_PERMISSION, correctHash)) {
            logLine(" The in-house signature permission is not d
eclared by in-house application.");
            return;
        }
        // *** POINT 10 *** Verify if the destination applicatio
n is signed with the in-house certificate.
        String pkgname = providerPkgname(this, Address.CONTENT_U
RI);
        if (!PkgCert.test(this, pkgname, correctHash)) {
            logLine(" The target content provider is not served
by in-house applications.");
            return;
        }
        // *** POINT 11 *** Sensitive information can be sent si
nce the destination application is in-house one.
        ContentValues values = new ContentValues();
        values.put("city", "Tokyo");
        String where = "_id = ?";
        String[] args = { "4" };
        int count = getContentResolver().update(Address.CONTENT_
URI, values, where, args);
        // *** POINT 12 *** Handle the received result data care
fully and securely,
        // even though the data comes from an in-house applicati
on.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        logLine(String.format(" %s records updated", count));
    }

    public void onDeleteClick(View view) {
        logLine("[Delete]");
        // *** POINT 9 *** Verify if the in-house signature perm
```

```
ission is defined by an in-house application.
        String correctHash = myCertHash(this);
        if (!SigPerm.test(this, MY_PERMISSION, correctHash)) {
            logLine(" The target content provider is not served
by in-house applications.");
            return;
        }
        // *** POINT 10 *** Verify if the destination applicatio
n is signed with the in-house certificat
        e.
        String pkgname = providerPkgname(this, Address.CONTENT_U
RI);
        if (!PkgCert.test(this, pkgname, correctHash)) {
            logLine(" The target content provider is not served
by in-house applications.");
            return;
        }
        // *** POINT 11 *** Sensitive information can be sent si
nce the destination application is in-ho
        use one.
        int count = getContentResolver().delete(Address.CONTENT_
URI, null, null);
        // *** POINT 12 *** Handle the received result data care
fully and securely,
        // even though the data comes from an in-house applicati
on.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        logLine(String.format(" %s records deleted", count));
    }

    private TextView mLogView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mLogView = (TextView)findViewById(R.id.logview);
    }

    private void logLine(String line) {
        mLogView.append(line);
        mLogView.append("¥n");
    }
}
```

SigPerm.java

```
package org.jssec.android.shared;

import android.content.Context;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.PermissionInfo;

public class SigPerm {

    public static boolean test(Context ctx, String sigPermName, String correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, sigPermName));
    }

    public static String hash(Context ctx, String sigPermName) {
        if (sigPermName == null) return null;
        try {
            // Get the package name of the application which declares a permission named sigPermName.
            PackageManager pm = ctx.getPackageManager();
            PermissionInfo pi;
            pi = pm.getPermissionInfo(sigPermName, PackageManager.GET_META_DATA);
            String pkgname = pi.packageName;
            // Fail if the permission named sigPermName is not a Signature Permission
            if (pi.protectionLevel != PermissionInfo.PROTECTION_SIGNATURE) return null;
            // Return the certificate hash value of the application which declares a permission named sigPermName.
            return PkgCert.hash(ctx, pkgname);
        } catch (NameNotFoundException e) {
            return null;
        }
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;
```

```
public class PkgCert {

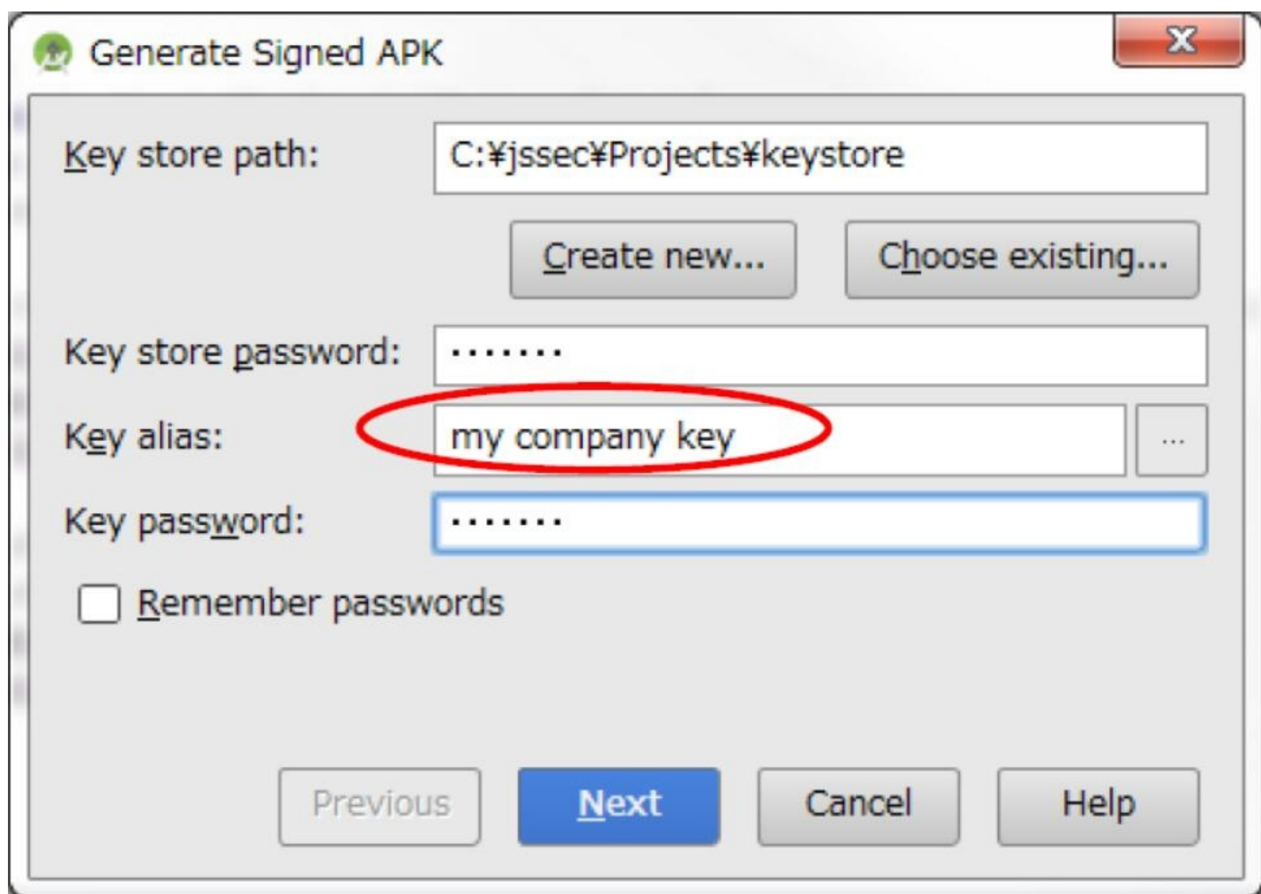
    public static boolean test(Context ctx, String pkgname, Stri
ng correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
    }
}
```

要点 13：导出 APK 时，请使用与请求应用相同的开发者密钥对 APK 进行签名。


Generate Signed APK
Key store path:
C:¥jssec¥Projects¥keystore
Create new...
Choose existing...
Key store password:
Key alias:
my company key
Key password:
Remember passwords
Previous
Next
Cancel
Help

4.3.1.5 创建/使用临时内容供应器

临时内容供应器基本上是一个私有内容供应器，但它允许特定的应用访问特定的 URI。通过向目标应用发送一个指定了特殊标志的意图，即可为这些应用提供临时访问权限。内容供应器方的应用可以将访问权限主动授予其他应用，并且还可以将访问权限被动授予索要临时访问权限的应用。

下面展示了实现临时内容供应器的示例代码。

要点（创建内容供应器）：

1) 将导出属性显式设置为 `false` 。

2) 使用 `grant-uri-permission` 指定路径来临时授予访问权。

3) 即使数据来自临时访问应用，也应该消息并安全地处理收到的请求数据。

4) 可以返回公开给临时访问应用的信息。

5) 为意图指定 URI 来授予临时访问权。

6) 为意图指定访问权限来授予临时访问权。

7) 将显式意图发送给应用来授予临时访问权。

8) 将意图返回给请求临时访问权的应用。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.provider.temporaryprovider">
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".TemporaryActiveGrantActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
        <!-- Temporary Content Provider -->
        <!-- *** POINT 1 *** Explicitly set the exported attribu
te to false. -->
        <provider
            android:name=".TemporaryProvider"
            android:authorities="org.jssec.android.provider.temp
oraryprovider"
            android:exported="false" >
            <!-- *** POINT 2 *** Specify the path to grant acces
s temporarily with the grant-uri-permissi
            on. -->
            <grant-uri-permission android:path="/addresses" />
        </provider>
        <activity
        android:name=".TemporaryPassiveGrantActivity"
        android:label="@string/app_name"
        android:exported="true" />
    </application>
</manifest>
```

TemporaryProvider.java

```
package org.jssec.android.provider.temporaryprovider;

import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.MatrixCursor;
import android.net.Uri;
```

```
public class TemporaryProvider extends ContentProvider {

    public static final String AUTHORITIY = "org.jssec.android.p
rovider.temporaryprovider";
    public static final String CONTENT_TYPE = "vnd.android.curso
r.dir/vnd.org.jssec.contenttype";
    public static final String CONTENT_ITEM_TYPE = "vnd.android.
cursor.item/vnd.org.jssec.contenttype";
    // Expose the interface that the Content Provider provides.

    public interface Download {
        public static final String PATH = "downloads";
        public static final Uri CONTENT_URI = Uri.parse("content
://" + AUTHORITIY + "/" + PATH);
    }

    public interface Address {
        public static final String PATH = "addresses";
        public static final Uri CONTENT_URI = Uri.parse("content
://" + AUTHORITIY + "/" + PATH);
    }

    // UriMatcher
    private static final int DOWNLOADS_CODE = 1;
    private static final int DOWNLOADS_ID_CODE = 2;
    private static final int ADDRESSES_CODE = 3;
    private static final int ADDRESSES_ID_CODE = 4;
    private static UriMatcher sUriMatcher;

    static {
        sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        sUriMatcher.addURI(AUTHORITIY, Download.PATH, DOWNLOADS_
CODE);
        sUriMatcher.addURI(AUTHORITIY, Download.PATH + "/#", DOW
NLOADS_ID_CODE);
        sUriMatcher.addURI(AUTHORITIY, Address.PATH, ADDRESSES_C
ODE);
        sUriMatcher.addURI(AUTHORITIY, Address.PATH + "/#", ADDR
ESSES_ID_CODE);
    }

    // Since this is a sample program,
    // query method returns the following fixed result always wi
thout using database.
    private static MatrixCursor sAddressCursor = new MatrixCurso
r(new String[] { "_id", "city" });

    static {
        sAddressCursor.addRow(new String[] { "1", "New York" });
        sAddressCursor.addRow(new String[] { "2", "London" });
        sAddressCursor.addRow(new String[] { "3", "Paris" });
    }
```

```
    private static MatrixCursor sDownloadCursor = new MatrixCurs
or(new String[] { "_id", "path" });

    static {
        sDownloadCursor.addRow(new String[] { "1", "/sdcard/down
loads/sample.jpg" });
        sDownloadCursor.addRow(new String[] { "2", "/sdcard/down
loads/sample.txt" });
    }

    @Override
    public boolean onCreate() {
        return true;
    }

    @Override
    public String getType(Uri uri) {
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
            case ADDRESSES_CODE:
                return CONTENT_TYPE;
            case DOWNLOADS_ID_CODE:
            case ADDRESSES_ID_CODE:
                return CONTENT_ITEM_TYPE;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String sel
ection,
        String[] selectionArgs, String sortOrder) {
        // *** POINT 3 *** Handle the received request data care
fully and securely,
        // even though the data comes from the application grant
ed access temporarily.
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Please refer to "3.2 Handle Input Data Carefully and
Securely."
        // *** POINT 4 *** Information that is granted to disclo
se to the temporary access applications can be returned.
        // It depends on application whether the query result ca
n be disclosed or not.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
            case DOWNLOADS_ID_CODE:
                return sDownloadCursor;
            case ADDRESSES_CODE:
```

```
            case ADDRESSES_ID_CODE:
                return sAddressCursor;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        // *** POINT 3 *** Handle the received request data care
fully and securely,
        // even though the data comes from the application grant
ed access temporarily.
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Please refer to "3.2 Handle Input Data Carefully and
Securely."
        // *** POINT 4 *** Information that is granted to disclo
se to the temporary access applications c
        an be returned.
        // It depends on application whether the issued ID has s
ensitive meaning or not.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
                return ContentUris.withAppendedId(Download.CONTE
NT_URI, 3);
            case ADDRESSES_CODE:
                return ContentUris.withAppendedId(Address.CONTEN
T_URI, 4);
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public int update(Uri uri, ContentValues values, String sele
ction,
        String[] selectionArgs) {
        // *** POINT 3 *** Handle the received request data care
fully and securely,
        // even though the data comes from the application grant
ed access temporarily.
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Please refer to "3.2 Handle Input Data Carefully and
Securely."
        // *** POINT 4 *** Information that is granted to disclo
```

```
se to the temporary access applications can be returned.
        // It depends on application whether the number of updat
ed records has sensitive meaning or not.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
                return 5; // Return number of updated records
            case DOWNLOADS_ID_CODE:
                return 1;
            case ADDRESSES_CODE:
                return 15;
            case ADDRESSES_ID_CODE:
                return 1;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }

    @Override
    public int delete(Uri uri, String selection, String[] select
ionArgs) {
        // *** POINT 3 *** Handle the received request data care
fully and securely,
        // even though the data comes from the application grant
ed access temporarily.
        // Here, whether uri is within expectations or not, is v
erified by UriMatcher#match() and switch case.
        // Checking for other parameters are omitted here, due t
o sample.
        // Please refer to "3.2 Handle Input Data Carefully and
Securely."
        // *** POINT 4 *** Information that is granted to disclo
se to the temporary access applications can be returned.
        // It depends on application whether the number of delet
ed records has sensitive meaning or not.
        switch (sUriMatcher.match(uri)) {
            case DOWNLOADS_CODE:
                return 10; // Return number of deleted records
            case DOWNLOADS_ID_CODE:
                return 1;
            case ADDRESSES_CODE:
                return 20;
            case ADDRESSES_ID_CODE:
                return 1;
            default:
                throw new IllegalArgumentException("Invalid URI:"
 + uri);
        }
    }
}
```

TemporaryActiveGrantActivity.java

```
package org.jssec.android.provider.temporaryprovider;

import android.app.Activity;
import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class TemporaryActiveGrantActivity extends Activity {

    // User Activity Information
    private static final String TARGET_PACKAGE = "org.jssec.andr
oid.provider.temporaryuser";
    private static final String TARGET_ACTIVITY = "org.jssec.and
roid.provider.temporaryuser.TemporaryUserActivity";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.active_grant);
    }

    // In the case that Content Provider application grants acce
ss permission to other application actively.
    public void onSendClick(View view) {
        try {
            Intent intent = new Intent();
            // *** POINT 5 *** Specify URI for the intent to gra
nt temporary access.
            intent.setData(TemporaryProvider.Address.CONTENT_URI
);
            // *** POINT 6 *** Specify access rights for the int
ent to grant temporary access.
            intent.setFlags(Intent.FLAG_GRANT_READ_URI_PERMISSIO
N);
            // *** POINT 7 *** Send the explicit intent to an ap
plication to grant temporary access.
            intent.setClassName(TARGET_PACKAGE, TARGET_ACTIVITY)
;
            startActivity(intent);
        } catch (ActivityNotFoundException e) {
            Toast.makeText(this, "User Activity not found.", Toa
st.LENGTH_LONG).show();
        }
    }
}
```

TemporaryPassiveGrantActivity.java

```
package org.jssec.android.provider.temporaryprovider;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class TemporaryPassiveGrantActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.passive_grant);
    }

    // In the case that Content Provider application passively g
rants access permission
    // to the application that requested Content Provider access.

    public void onGrantClick(View view) {
        Intent intent = new Intent();
        // *** POINT 5 *** Specify URI for the intent to grant t
emporary access.
        intent.setData(TemporaryProvider.Address.CONTENT_URI);
        // *** POINT 6 *** Specify access rights for the intent
to grant temporary access.
        intent.setFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
        // *** POINT 8 *** Return the intent to the application
that requests temporary access.
        setResult(Activity.RESULT_OK, intent);
        finish();
    }

    public void onCloseClick(View view) {
        finish();
    }
}
```

下面是临时内容供应器的示例。

要点（使用内容供应器）：

9) 不要发送敏感信息。

10) 收到结果时，小心并安全地处理结果数据。

TemporaryUserActivity.java

```
package org.jssec.android.provider.temporaryuser;
```

```
import android.app.Activity;
import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.content.pm.ProviderInfo;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class TemporaryUserActivity extends Activity {

    // Information of the Content Provider's Activity to request
 temporary content provider access.
    private static final String TARGET_PACKAGE = "org.jssec.andr
oid.provider.temporaryprovider";
    private static final String TARGET_ACTIVITY = "org.jssec.and
roid.provider.temporaryprovider.TemporaryPassiveGrantActivity";
    // Target Content Provider Information
    private static final String AUTHORITY = "org.jssec.android.p
rovider.temporaryprovider";

    private interface Address {
        public static final String PATH = "addresses";
        public static final Uri CONTENT_URI = Uri.parse("content
://" + AUTHORITY + "/" + PATH);
    }

    private static final int REQUEST_CODE = 1;

    public void onQueryClick(View view) {
        logLine("[Query]");
        Cursor cursor = null;
        try {
            if (!providerExists(Address.CONTENT_URI)) {
                logLine(" Content Provider doesn't exist.");
                return;
            }
            // *** POINT 9 *** Do not send sensitive information.

            // If no problem when the information is taken by ma
lware, it can be included in the request.
            cursor = getContentResolver().query(Address.CONTENT_
URI, null, null, null, null);
            // *** POINT 10 *** When receiving a result, handle
the result data carefully and securely.
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
            if (cursor == null) {
                logLine(" null cursor");
            } else {
                boolean moved = cursor.moveToFirst();
                while (moved) {
```

```
                    logLine(String.format(" %d, %s", cursor.getI
nt(0), cursor.getString(1)));
                    moved = cursor.moveToNext();
                }
            }
        } catch (SecurityException ex) {
            logLine(" Exception:" + ex.getMessage());
        }
        finally {
            if (cursor != null) cursor.close();
        }
    }

    // In the case that this application requests temporary acce
ss to the Content Provider
    // and the Content Provider passively grants temporary acces
s permission to this application.
    public void onGrantRequestClick(View view) {
        Intent intent = new Intent();
        intent.setClassName(TARGET_PACKAGE, TARGET_ACTIVITY);
        try {
            startActivityForResult(intent, REQUEST_CODE);
        } catch (ActivityNotFoundException e) {
            logLine("Content Provider's Activity not found.");
        }
    }

    private boolean providerExists(Uri uri) {
        ProviderInfo pi = getPackageManager().resolveContentProv
ider(uri.getAuthority(), 0);
        return (pi != null);
    }

    private TextView mLogView;

    // In the case that the Content Provider application grants
temporary access
    // to this application actively.
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mLogView = (TextView)findViewById(R.id.logview);
    }

    private void logLine(String line) {
        mLogView.append(line);
        mLogView.append("¥n");
    }
}
```

4.3.2 规则书

实现或使用内容供应器时，确保遵循以下规则。

4.3.2.1 仅仅在应用中使用的内容供应器必须设为私有（必需）

仅供单个应用使用的内容供应器不需要被其他应用访问，并且开发人员通常不会考虑攻击内容供应器的访问。 内容供应器基本上是共享数据的系统，因此它默认处理成公共的。 仅在单个应用中使用的内容供应器应该被显式设置为私有，并且它应该是私有内容供应器。 在 Android 2.3.1（API Level 9）或更高版本中，通过在 `provider` 元素中指定 `android:exported="false"` ，可以将内容供应器设置为私有。

AndroidManifest.xml

```
<!-- *** POINT 1 *** Set false for the exported attribute explic
itly. -->
<provider
android:name=".PrivateProvider"
android:authorities="org.jssec.android.provider.privateprovider"
android:exported="false" />
```

4.3.2.2 小心并安全地处理收到的请求参数（必需）

风险因内容供应器的类型而异，但在处理请求参数时，你应该做的第一件事是输入验证。

虽然内容供应器的每个方法，都有一个接口，应该接收 SQL 语句的成分参数，但实际上它只是交给系统中的任意字符串，所以需要注意内容供应器方需要假设，可能会提供意外的参数的情况。

由于公共内容提供应器可以接收来自不受信任来源的请求，因此可能会受到恶意软件的攻击。 另一方面，私有内容供应器永远不会直接收到来自其他应用的任何请求，但是目标应用中的公共活动，可能会将恶意意图转发给私有内容供应器，因此你不应该认为，私有内容供应器不能 接收任何恶意输入。 由于其他内容供应器也有将恶意意图转发给他们的风险，因此有必要对这些请求执行输入验证。

请参阅“3.2 小心和安全地处理输入数据”。

4.3.2.3 验证签名权限由内部定义之后，使用内部定义的签名权限（必需）

确保在创建内容供应器时，通过定义内部签名权限，来保护你的内部内容供应器。 由于在 `AndroidManifest.xml` 文件中定义权限或声明权限请求，没有提供足够的安全性，请务必参考“5.2.1.2 如何使用内部定义的签名权限在内部应用之间进行通

信”。

4.3.2.4 返回结果时，请注意来自目标应用的结果的信息泄露的可能性（必须）

在 `query()` 或插入 `insert()` 的情况下， `Cursor` 或 `Uri` 作为结果信息返回到发送请求的应用。 当敏感信息包含在结果信息中时，信息可能会从目标应用泄露。 在 `update()` 或 `delete()` 的情况下，更新/删除记录的数量作为结果信息返回给发送请求的应用。 在极少数情况下，取决于某些应用的规范，更新/删除记录的数量具有敏感含义，请注意这一点。

4.3.2.5 提供二手素材时，素材应该以相同级别的保护提供（必需）

当受到权限保护的信息或功能素材，被另一个应用提供时，你需要确保它具有访问素材所需的相同权限。 在 Android OS 权限安全模型中，只有已被授予适当权限的应用，才能直接访问受保护的素材。 但是，存在一个漏洞，因为具有素材权限的应用可以充当代理，并允许非特权应用的访问。 基本上这与重授权限相同，因此它被称为“重新授权”问题。 请参阅“5.2.3.4 重新授权问题”。

4.3.2.6 小心并安全地处理来自内容供应器的返回的结果数据（必需）

风险因内容供应器的类型而异，但在处理请求参数时，你应该做的第一件事是输入验证。

如果目标内容供应器是公共内容供应器，伪装成公共内容供应器的恶意软件可能会返回攻击性结果数据。 另一方面，如果目标内容供应器是私有内容供应器，则其风险较小，因为它从同一应用接收结果数据，但不应该认为，私有内容供应器不能接收任何恶意输入。 由于其他内容供应器也有将恶意数据返回给他们的风险，因此有必要对该结果数据执行输入验证。

请参阅“3.2 小心和安全地处理输入数据”。

4.4 创建/使用服务

4.4.1 示例代码

使用服务的风险和对策取决于服务的使用方式。 您可以通过下面展示的图表找出您应该创建的服务类型。 由于安全编码的最佳实践，根据服务的创建方式而有所不同，因此我们也将解释服务的实现。

表 4.4-1 服务类型的定义

类型	定义
私有	不能由其他应用加载，所以是最安全的服务
公共	应该由很多未指定的应用使用的服务
伙伴	只能由可信的伙伴公司开发的应用使用的服务
内部	只能由其他内部应用使用的服务

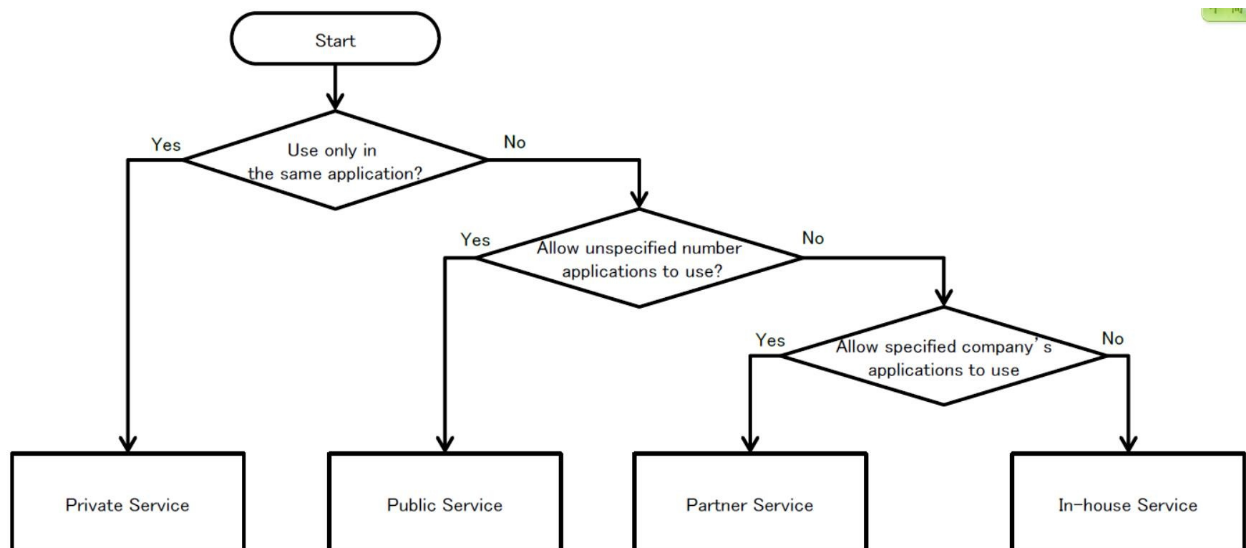


Figure 4.4-1

有几种服务实现方法，您将选择匹配您想要创建的服务类型的方法。 表中列的条目展示了实现方法，并将它们分为 5 种类型。 “OK”表示可能的组合，其他表示不可能/困难的组合。

服务的详细实现方法，请参阅“4.4.3.2 如何实现服务”和每个服务类型的示例代码（在表中带有`*`标记）。

表 4.4-2

类别	私有服务	公共服务	伙伴服务	内部服务
`startService` 类型	OK*	OK	-	OK
`IntentService` 类型	OK	OK*	-	OK
本地绑定类型	OK	-	-	-
`Messenger` 绑定类型	OK	OK	-	OK*
AIDL 绑定类型	OK	OK	OK*	OK

每种服务安全类型的示例代码展示在下面，通过表 4.4-2 中的使用 `*` 标记。

4.4.1.1 创建/使用私有服务

私有服务是不能由其他应用启动的服务，因此它是最安全的服务。 当使用仅在应用中使用的私有服务时，只要您对该类使用显式意图，那么您就不必担心意外将它发送到任何其他应用。

下面展示了如何使用`startService`类型服务的示例代码。

要点（创建服务）：

1) 将导出属性显式设置为`false`。

2) 小心并安全地处理收到的意图，即使意图从相同应用发送。

3) 由于请求应用在同一应用中，所以可以发送敏感信息。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.service.privateservice" >
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:allowBackup="false" >
        <activity
            android:name=".PrivateUserActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
        <!-- Private Service derived from Service class -->
        <!-- *** POINT 1 *** Explicitly set the exported attribu
te to false. -->
        <service android:name=".PrivateStartService" android:exp
orted="false"/>
        <!-- Private Service derived from IntentService class -->

        <!-- *** POINT 1 *** Explicitly set the exported attribu
te to false. -->
        <service android:name=".PrivateIntentService" android:ex
ported="false"/>
    </application>
</manifest>
```

PrivateStartService.java

```
package org.jssec.android.service.privateservice;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;

public class PrivateStartService extends Service {

    // The onCreate gets called only one time when the service s
tarts.
    @Override
    public void onCreate() {
        Toast.makeText(this, "PrivateStartService - onCreate()",
 Toast.LENGTH_SHORT).show();
    }

    // The onStartCommand gets called each time after the startS
ervice gets called.
    @Override
    public int onStartCommand(Intent intent, int flags, int star
tId) {
        // *** POINT 2 *** Handle the received intent carefully
and securely,
        // even though the intent was sent from the same applica
tion.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        String param = intent.getStringExtra("PARAM");
        Toast.makeText(this,
        String.format("PrivateStartService¥nReceived param: ¥"%s¥
"", param),
        Toast.LENGTH_LONG).show();
        return Service.START_NOT_STICKY;
    }

    // The onDestroy gets called only one time when the service
stops.
    @Override
    public void onDestroy() {
        Toast.makeText(this, "PrivateStartService - onDestroy()"
, Toast.LENGTH_SHORT).show();
    }

    @Override
    public IBinder onBind(Intent intent) {
        // This service does not provide binding, so return null
        return null;
    }
}
```

下面是使用私有服务的活动代码：

要点（使用服务）：

4) 使用指定类的显式意图，调用同一应用程序的服务。

5) 由于目标服务位于同一应用中，因此可以发送敏感信息。

6) 即使数据来自同一应用中的服务，也要小心并安全地处理收到的结果数据。

PrivateUserActivity.java

```
package org.jssec.android.service.privateservice;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class PrivateUserActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.privateservice_activity);
    }

    // --- StartService control ---
    public void onStartServiceClick(View v) {
        // *** POINT 4 *** Use the explicit intent with class sp
ecified to call a service in the same application.
        Intent intent = new Intent(this, PrivateStartService.cla
ss);
        // *** POINT 5 *** Sensitive information can be sent sin
ce the destination service is in the same application.
        intent.putExtra("PARAM", "Sensitive information");
        startService(intent);
    }

    public void onStopServiceClick(View v) {
        doStopService();
    }

    @Override
    public void onStop() {
        super.onStop();
        // Stop service if the service is running.
        doStopService();
    }

    private void doStopService() {
        // *** POINT 4 *** Use the explicit intent with class sp
ecified to call a service in the same application.
```

```
        Intent intent = new Intent(this, PrivateStartService.cla
ss);
        stopService(intent);
    }

    // --- IntentService control ---
    public void onIntentServiceClick(View v) {
        // *** POINT 4 *** Use the explicit intent with class sp
ecified to call a service in the same application.
        Intent intent = new Intent(this, PrivateIntentService.cl
ass);
        // *** POINT 5 *** Sensitive information can be sent sin
ce the destination service is in the same application.
        intent.putExtra("PARAM", "Sensitive information");
        startService(intent);
    }
}
```

4.4.1.2 创建/使用公共服务

公共服务是应该由未指定的大量应用使用的服务。 有必要注意，它可能会收到恶意软件发送的信息（意图等）。 在使用公共服务的情况下，有必要注意，恶意软件可能会收到要发送的信息（意图等）。

下面展示了如何使用 `startService` 类型服务的示例代码。

要点（创建服务）：

1) 将导出属性显式设置为 `true` 。

2) 小心并安全地处理接收到的意图。

3) 返回结果时，请勿包含敏感信息。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.service.publicservice" >
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:allowBackup="false" >
        <!-- Most standard Service -->
        <!-- *** POINT 1 *** Explicitly set the exported attribu
te to true. -->
        <service android:name=".PublicStartService" android:expo
rted="true">
            <intent-filter>
                <action android:name="org.jssec.android.service.
publicservice.action.startservice" />
            </intent-filter>
        </service>
        <!-- Public Service derived from IntentService class -->
        <!-- *** POINT 1 *** Explicitly set the exported attribu
te to true. -->
        <service android:name=".PublicIntentService" android:exp
orted="true">
            <intent-filter>
                <action android:name="org.jssec.android.service.
publicservice.action.intentservice" />
            </intent-filter>
        </service>
    </application>
</manifest>
```

PublicIntentService.java

```
package org.jssec.android.service.publicservice;

import android.app.IntentService;
import android.content.Intent;
import android.widget.Toast;

public class PublicIntentService extends IntentService{

    /**
    * Default constructor must be provided when a service extend
s IntentService class.
    * If it does not exist, an error occurs.
    */
    public PublicIntentService() {
        super("CreatingTypeBService");
    }

    // The onCreate gets called only one time when the Service s
tarts.
    @Override
    public void onCreate() {
        super.onCreate();
        Toast.makeText(this, this.getClass().getSimpleName() + "
 - onCreate()", Toast.LENGTH_SHORT).show();
    }

    // The onHandleIntent gets called each time after the startS
ervice gets called.
    @Override
    protected void onHandleIntent(Intent intent) {
        // *** POINT 2 *** Handle intent carefully and securely.
        // Since it's public service, the intent may come from m
alicious application.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        String param = intent.getStringExtra("PARAM");
        Toast.makeText(this, String.format("Recieved parameter ¥"
%s¥"", param), Toast.LENGTH_LONG).show();
    }

    // The onDestroy gets called only one time when the service
stops.
    @Override
    public void onDestroy() {
        Toast.makeText(this, this.getClass().getSimpleName() + "
 - onDestroy()", Toast.LENGTH_SHORT).show();
    }
}
```

下面是使用公共服务的活动代码：

要点（使用服务）：

4) 不要发送敏感信息。

5) 收到结果时，小心并安全地处理结果数据。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.service.publicserviceuser" >
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:allowBackup="false" >
        <activity
            android:name=".PublicUserActivity"
            android:label="@string/app_name"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

PublicUserActivity.java

```
package org.jssec.android.service.publicserviceuser;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class PublicUserActivity extends Activity {

    // Using Service Info
    private static final String TARGET_PACKAGE = "org.jssec.andr
oid.service.publicservice";
    private static final String TARGET_START_CLASS = "org.jssec.
android.service.publicservice.PublicStartService";
    private static final String TARGET_INTENT_CLASS = "org.jssec
```

```
.android.service.publicservice.PublicIntentService";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.publicservice_activity);
    }

    // --- StartService control ---
    public void onStartServiceClick(View v) {
        Intent intent = new Intent("org.jssec.android.service.pu
blicservice.action.startservice");
        // *** POINT 4 *** Call service by Explicit Intent
        intent.setClassName(TARGET_PACKAGE, TARGET_START_CLASS);
        // *** POINT 5 *** Do not send sensitive information.
        intent.putExtra("PARAM", "Not sensitive information");
        startService(intent);
        // *** POINT 6 *** When receiving a result, handle the r
esult data carefully and securely.
        // This sample code uses startService(), so receiving no
 result.
    }

    public void onStopServiceClick(View v) {
        doStopService();
    }

    // --- IntentService control ---
    public void onIntentServiceClick(View v) {
        Intent intent = new Intent("org.jssec.android.service.pu
blicservice.action.intentservice");
        // *** POINT 4 *** Call service by Explicit Intent
        intent.setClassName(TARGET_PACKAGE, TARGET_INTENT_CLASS)
;
        // *** POINT 5 *** Do not send sensitive information.
        intent.putExtra("PARAM", "Not sensitive information");
        startService(intent);
    }

    @Override
    public void onStop(){
        super.onStop();
        // Stop service if the service is running.
        doStopService();
    }

    // Stop service
    private void doStopService() {
        Intent intent = new Intent("org.jssec.android.service.pu
blicservice.action.startservice");
        // *** POINT 4 *** Call service by Explicit Intent
        intent.setClassName(TARGET_PACKAGE, TARGET_START_CLASS);
        stopService(intent);
```

```
        }
}
```

4.4.1.3 创建/使用伙伴服务

伙伴服务是只能由特定应用使用的服务。 系统由伙伴公司的应用和内部应用组成，用于保护在伙伴应用和内部应用之间处理的信息和功能。

以下是 AIDL 绑定类型服务的示例。

要点（创建服务）：

1) 不要定义意图过滤器，并将导出属性显式设置为 `true` 。

2) 验证请求应用的证书是否已在自己的白名单中注册。

3) 请勿（无法）通过 `onBind(onStartCommand, onHandleIntent)` 识别请求应用是否为伙伴。

4) 小心并安全地处理接收到的意图，即使意图是从伙伴应用发送的。

5) 仅返回公开给伙伴应用的信息。

另外，请参阅“5.2.1.3 如何验证应用证书的哈希值”，来了解如何验证目标应用的哈希值，它在白名单中指定。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.service.partnerservice.aidl" >
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:allowBackup="false" >
        <!-- Service using AIDL -->
        <!-- *** POINT 1 *** Do not define the intent filter and
 explicitly set the exported attribute to
        true. -->
        <service
        android:name="org.jssec.android.service.partnerservice.a
idl.PartnerAIDLService"
        android:exported="true" />
    </application>
</manifest>
```

在这个例子中，将创建 2 个 AIDL 文件。 一个是回调接口，将数据从服务提供给活动。 另一个接口将数据从活动提供给服务，并获取信息。 另外，AIDL 文件中描述的包名称，应与 AIDL 文件的目录层次一致，与 `java` 文件中描述的包名称相同。

IExclusiveAIDLServiceCallback.aidl

```
package org.jssec.android.service.exclusiveservice.aidl;

interface IExclusiveAIDLServiceCallback {

    /**
    * It's called when the value is changed.
    */
    void valueChanged(String info);
}
```

IExclusiveAIDLService.aidl

```
package org.jssec.android.service.exclusiveservice.aidl;

import org.jssec.android.service.exclusiveservice.aidl.IExclusiv
eAIDLServiceCallback;

interface IExclusiveAIDLService {

    /**
    * Register Callback.
    */
    void registerCallback(IExclusiveAIDLServiceCallback cb);

    /**
    * Get Information
    */
    String getInfo(String param);

    /**
    * Unregister Callback
    */
    void unregisterCallback(IExclusiveAIDLServiceCallback cb);
}
```

PartnerAIDLService.java

```
package org.jssec.android.service.partnerservice.aidl;

import org.jssec.android.shared.PkgCertWhitelists;
import org.jssec.android.shared.Utils;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.os.Handler;
import android.os.IBinder;
import android.os.Message;
import android.os.RemoteCallbackList;
import android.os.RemoteException;
```

```
import android.widget.Toast;

public class PartnerAIDLService extends Service {

    private static final int REPORT_MSG = 1;
    private static final int GETINFO_MSG = 2;
    // The value which this service informs to client
    private int mValue = 0;
    // *** POINT 2 *** Verify that the certificate of the reques
ting application has been registered in the own white list.
    private static PkgCertWhitelists sWhitelists = null;

    private static void buildWhitelists(Context context) {
        boolean isdebug = Utils.isDebuggable(context);
        sWhitelists = new PkgCertWhitelists();
        // Register certificate hash value of partner applicatio
n "org.jssec.android.service.partnerservice.aidluser"
        sWhitelists.add("org.jssec.android.service.partnerservic
e.aidluser", isdebug ?
        // Certificate hash value of debug.keystore "androiddebu
gkey"
            "0EFB7236 328348A9 89718BAD DF57F544 D5CCB4AE B9DB34
BC 1E29DD26 F77C8255" :
            // Certificate hash value of keystore "partner key"
            "1F039BB5 7861C27A 3916C778 8E78CE00 690B3974 3EB825
9F E2627B8D 4C0EC35A");
            // Register other partner applications in the same w
ay
    }

    private static boolean checkPartner(Context context, String
pkgname) {
        if (sWhitelists == null) buildWhitelists(context);
        return sWhitelists.test(context, pkgname);
    }

    // Object to register callback
    // Methods which RemoteCallbackList provides are thread-safe.

    private final RemoteCallbackList<IPartnerAIDLServiceCallback
> mCallbacks =
        new RemoteCallbackList<IPartnerAIDLServiceCallback>();

    // Handler to send data when callback is called.
    private static class ServiceHandler extends Handler{
    private Context mContext;
    private RemoteCallbackList<IPartnerAIDLServiceCallback> mCal
lbacks;
    private int mValue = 0;

    public ServiceHandler(Context context, RemoteCallbackList<IP
artnerAIDLServiceCallback> callback, int value){
        this.mContext = context;
```

```
        this.mCallbacks = callback;
        this.mValue = value;
    }

    @Override
    public void handleMessage(Message msg) {
        switch (msg.what) {
            case REPORT_MSG: {
                if(mCallbacks == null){
                    return;
                }
                // Start broadcast
                // To call back on to the registered clients, us
e beginBroadcast().
                // beginBroadcast() makes a copy of the currentl
y registered callback list.
                final int N = mCallbacks.beginBroadcast();
                for (int i = 0; i < N; i++) {
                    IPartnerAIDLServiceCallback target = mCallba
cks.getBroadcastItem(i);
                    try {
                        // *** POINT 5 *** Information that is g
ranted to disclose to partner applications can be returned.
                        target.valueChanged("Information disclos
ed to partner application (callback from Service) No." + (++mVal
ue));
                    } catch (RemoteException e) {
                        // Callbacks are managed by RemoteCallba
ckList, do not unregister callbacks here.
                        // RemoteCallbackList.kill() unregister
all callbacks
                    }
                }
                // finishBroadcast() cleans up the state of a br
oadcast previously initiated by calling beginBroadcast().
                mCallbacks.finishBroadcast();
                // Repeat after 10 seconds
                sendEmptyMessageDelayed(REPORT_MSG, 10000);
                break;
            }
            case GETINFO_MSG: {
                if(mContext != null) {
                    Toast.makeText(mContext,
                        (String) msg.obj, Toast.LENGTH_LONG).sho
w();
                }
                break;
            }
            default:
                super.handleMessage(msg);
                break;
        } // switch
    }
```

```
    protected final ServiceHandler mHandler = new ServiceHandler(
this, mCallbacks, mValue);

    // Interfaces defined in AIDL
    private final IPartnerAIDLService.Stub mBinder = new IPartne
rAIDLService.Stub() {
        private boolean checkPartner() {
            Context ctx = PartnerAIDLService.this;
            if (!PartnerAIDLService.checkPartner(ctx, Utils.getP
ackageNameFromUid(ctx, getCallingUid()))) {
                mHandler.post(new Runnable(){
                    @Override
                    public void run(){
                        Toast.makeText(PartnerAIDLService.this,
"Requesting application is not partner application.", Toast.LENG
TH_LONG).show();
                    }
                });
                return false;
            }
            return true;
        }

        public String getInfo(String param) {
            // *** POINT 2 *** Verify that the certificate of th
e requesting application has been registered in the own white li
st.
            if (!checkPartner()) {
                return null;
            }
            // *** POINT 4 *** Handle the received intent carefu
lly and securely,
            // even though the intent was sent from a partner ap
plication
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
            Message msg = new Message();
            msg.what = GETINFO_MSG;
            msg.obj = String.format("Method calling from partner
 application. Recieved ¥"%s¥"", param);
            PartnerAIDLService.this.mHandler.sendMessage(msg);
            // *** POINT 5 *** Return only information that is g
ranted to be disclosed to a partner application.
            return "Information disclosed to partner application
 (method from Service)";
        }

        public void unregisterCallback(IPartnerAIDLServiceCallba
ck cb) {
            // *** POINT 2 *** Verify that the certificate of th
e requesting application has been registered in the own white li
st.
```

```
            if (!checkPartner()) {
                return;
            }
            if (cb != null) mCallbacks.unregister(cb);
        }
    };

    @Override
    public IBinder onBind(Intent intent) {
        // *** POINT 3 *** Verify that the certificate of the re
questing application has been registered in the own white list.
        // So requesting application must be validated in method
s defined in AIDL every time.
        return mBinder;
    }

    @Override
    public void onCreate() {
        Toast.makeText(this, this.getClass().getSimpleName() + "
 - onCreate()", Toast.LENGTH_SHORT).show();
        // During service is running, inform the incremented num
ber periodically.
        mHandler.sendEmptyMessage(REPORT_MSG);
    }

    @Override
    public void onDestroy() {
        Toast.makeText(this, this.getClass().getSimpleName() + "
 - onDestroy()", Toast.LENGTH_SHORT).show();
        // Unregister all callbacks
        mCallbacks.kill();
        mHandler.removeMessages(REPORT_MSG);
    }
}
```

PkgCertWhitelists.java

```
package org.jssec.android.shared;

import java.util.HashMap;
import java.util.Map;
import android.content.Context;

public class PkgCertWhitelists {

    private Map<String, String> mWhitelists = new HashMap<String
, String>();

    public boolean add(String pkgname, String sha256) {
        if (pkgname == null) return false;
        if (sha256 == null) return false;
        sha256 = sha256.replaceAll(" ", "");
        if (sha256.length() != 64) return false; // SHA-256 -> 3
2 bytes -> 64 chars
        sha256 = sha256.toUpperCase();
        if (sha256.replaceAll("[0-9A-F]+", "").length() != 0) re
turn false; // found non hex char
        mWhitelists.put(pkgname, sha256);
        return true;
    }

    public boolean test(Context ctx, String pkgname) {
        // Get the correct hash value which corresponds to pkgna
me.
        String correctHash = mWhitelists.get(pkgname);
        // Compare the actual hash value of pkgname with the cor
rect hash value.
        return PkgCert.test(ctx, pkgname, correctHash);
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;

public class PkgCert {

    public static boolean test(Context ctx, String pkgname, Stri
ng correctHash) {
        if (correctHash == null) return false;
```

```
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
    }
}
```

下面是使用伙伴服务的活动代码：

要点（使用服务）：

6) 验证目标应用的证书是否已在自己的白名单中注册。

7) 仅返回公开给伙伴应用的信息。

8) 使用显式意图调用伙伴服务。

9) 即使数据来自伙伴应用，也要小心并安全地处理收到的结果数据。

ExclusiveAIDLUserActivity.java

```java
package org.jssec.android.service.partnerservice.aidluser;

import org.jssec.android.service.partnerservice.aidl.IPartnerAID
LService;
import org.jssec.android.service.partnerservice.aidl.IPartnerAID
LServiceCallback;
import org.jssec.android.shared.PkgCertWhitelists;
import org.jssec.android.shared.Utils;
import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.Message;
import android.os.RemoteException;
import android.view.View;
import android.widget.Toast;

public class PartnerAIDLUserActivity extends Activity {

    private boolean mIsBound;
    private Context mContext;
    private final static int MGS_VALUE_CHANGED = 1;
    // *** POINT 6 *** Verify if the certificate of the target a
pplication has been registered in the own white list.
    private static PkgCertWhitelists sWhitelists = null;

    private static void buildWhitelists(Context context) {
        boolean isdebug = Utils.isDebuggable(context);
        sWhitelists = new PkgCertWhitelists();
        // Register certificate hash value of partner service ap
plication "org.jssec.android.service.partnerservice.aidl"
        sWhitelists.add("org.jssec.android.service.partnerservic
e.aidl", isdebug ?
            // Certificate hash value of debug.keystore "android
debugkey"
            "0EFB7236 328348A9 89718BAD DF57F544 D5CCB4AE B9DB34
BC 1E29DD26 F77C8255" :
            // Certificate hash value of keystore "my company ke
y"
            "D397D343 A5CBC10F 4EDDEB7C A10062DE 5690984F 1FB9E8
8B D7B3A7C2 42E142CA");
            // Register other partner service applications in th
e same way
    }

    private static boolean checkPartner(Context context, String
pkgname) {
        if (sWhitelists == null) buildWhitelists(context);
```

```
        return sWhitelists.test(context, pkgname);
    }

    // Information about destination (requested) partner activit
y.
    private static final String TARGET_PACKAGE = "org.jssec.andr
oid.service.partnerservice.aidl";
    private static final String TARGET_CLASS = "org.jssec.androi
d.service.partnerservice.aidl.PartnerAIDLService";

    private static class ReceiveHandler extends Handler{
        private Context mContext;
        public ReceiveHandler(Context context){
        this.mContext = context;
    }

    @Override
    public void handleMessage(Message msg) {
        switch (msg.what) {
            case MGS_VALUE_CHANGED: {
                String info = (String)msg.obj;
                Toast.makeText(mContext, String.format("Received
 ¥"%s¥" with callback.", info), Toast.LENGTH_SHORT).show();
                break;
            }
            default:
                super.handleMessage(msg);
                break;
        } // switch
    }

    private final ReceiveHandler mHandler = new ReceiveHandler(t
his);

    // Interfaces defined in AIDL. Receive notice from service
    private final IPartnerAIDLServiceCallback.Stub mCallback =
    new IPartnerAIDLServiceCallback.Stub() {
        @Override
        public void valueChanged(String info) throws RemoteExcep
tion {
            Message msg = mHandler.obtainMessage(MGS_VALUE_CHANG
ED, info);
            mHandler.sendMessage(msg);
        }
    };

    // Interfaces defined in AIDL. Inform service.
    private IPartnerAIDLService mService = null;
    // Connection used to connect with service. This is necessar
y when service is implemented with bindService().
    private ServiceConnection mConnection = new ServiceConnectio
n() {
        // This is called when the connection with the service h
```

```
as been established.
        @Override
        public void onServiceConnected(ComponentName className,
IBinder service) {
            mService = IPartnerAIDLService.Stub.asInterface(serv
ice);
            try{
                // connect to service
                mService.registerCallback(mCallback);
            }catch(RemoteException e){
                // service stopped abnormally
            }
            Toast.makeText(mContext, "Connected to service", Toa
st.LENGTH_SHORT).show();
        }

        // This is called when the service stopped abnormally an
d connection is disconnected.
        @Override
        public void onServiceDisconnected(ComponentName classNam
e) {
            Toast.makeText(mContext, "Disconnected from service"
, Toast.LENGTH_SHORT).show();
        }
    };

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.partnerservice_activity);
        mContext = this;
    }

    // --- StartService control ---
    public void onStartServiceClick(View v) {
        // Start bindService
        doBindService();
    }

    public void onGetInfoClick(View v) {
        getServiceinfo();
    }

    public void onStopServiceClick(View v) {
        doUnbindService();
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        doUnbindService();
    }
```

```
    /**
    * Connect to service
    */
    private void doBindService() {
        if (!mIsBound){
        // *** POINT 6 *** Verify if the certificate of the targ
et application has been registered in the own white list.
        if (!checkPartner(this, TARGET_PACKAGE)) {
            Toast.makeText(this, "Destination(Requested) sevice
application is not registered in white list.", Toast.LENGTH_LONG
).show();
            return;
        }
        Intent intent = new Intent();
        // *** POINT 7 *** Return only information that is grant
ed to be disclosed to a partner application.
        intent.putExtra("PARAM", "Information disclosed to partn
er application");
        // *** POINT 8 *** Use the explicit intent to call a par
tner service.
        intent.setClassName(TARGET_PACKAGE, TARGET_CLASS);
        bindService(intent, mConnection, Context.BIND_AUTO_CREAT
E);
        mIsBound = true;
        }
    }

    /**
    * Disconnect service
    */
    private void doUnbindService() {
        if (mIsBound) {
            // Unregister callbacks which have been registered.
            if(mService != null){
                try{
                    mService.unregisterCallback(mCallback);
                }catch(RemoteException e){
                    // Service stopped abnormally
                    // Omitted, since it' s sample.
                }
            }
            unbindService(mConnection);
            Intent intent = new Intent();
            // *** POINT 8 *** Use the explicit intent to call a
 partner service.
            intent.setClassName(TARGET_PACKAGE, TARGET_CLASS);
            stopService(intent);
            mIsBound = false;
        }
    }

    /**
    * Get information from service
```

```
    */
    void getServiceinfo() {
        if (mIsBound && mService != null) {
            String info = null;
            try {
                // *** POINT 7 *** Return only information that
is granted to be disclosed to a partner application.
                info = mService.getInfo("Information disclosed t
o partner application (method from activity)");
            } catch (RemoteException e) {
                e.printStackTrace();
            }
            // *** POINT 9 *** Handle the received result data c
arefully and securely,
            // even though the data came from a partner applicat
ion.
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
            Toast.makeText(mContext, String.format("Received ¥"%
s¥" from service.", info), Toast.LENGTH_SHORT).show();
        }
    }
}
```

PkgCertWhitelists.java

```
package org.jssec.android.shared;

import java.util.HashMap;
import java.util.Map;
import android.content.Context;

public class PkgCertWhitelists {

    private Map<String, String> mWhitelists = new HashMap<String
, String>();

    public boolean add(String pkgname, String sha256) {
        if (pkgname == null) return false;
        if (sha256 == null) return false;
        sha256 = sha256.replaceAll(" ", "");
        if (sha256.length() != 64) return false; // SHA-256 -> 3
2 bytes -> 64 chars
        sha256 = sha256.toUpperCase();
        if (sha256.replaceAll("[0-9A-F]+", "").length() != 0) re
turn false; // found non hex char
        mWhitelists.put(pkgname, sha256);
        return true;
    }

    public boolean test(Context ctx, String pkgname) {
        // Get the correct hash value which corresponds to pkgna
me.
        String correctHash = mWhitelists.get(pkgname);
        // Compare the actual hash value of pkgname with the cor
rect hash value.
        return PkgCert.test(ctx, pkgname, correctHash);
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;

public class PkgCert {

    public static boolean test(Context ctx, String pkgname, Stri
ng correctHash) {
        if (correctHash == null) return false;
```

```
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
    }
}
```

4.4.1.4 创建/使用内部服务

内部服务是除了内部应用以外的应用禁止使用的服务。 它们用于内部开发的应用，以便安全地共享信息和功能。 以下是使用 `Messenger` 绑定类型服务的示例。

要点（创建服务）：

1) 定义内部签名权限。

2) 需要内部签名权限。

3) 不要定义意图过滤器，并将导出属性显式设置为 `true` 。

4) 确认内部签名权限是由内部应用定义的。

5) 尽管意图是从内部应用发送的，但要小心并安全地处理接收到的意图。

6) 由于请求应用是内部的，因此可以返回敏感信息。

7) 导出 APK 时，请使用与请求应用相同的开发人员密钥对 APK 进行签名。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.service.inhouseservice.messenger"
 >
    <!-- *** POINT 1 *** Define an in-house signature permission
 -->
    <permission
    android:name="org.jssec.android.service.inhouseservice.messe
nger.MY_PERMISSION"
    android:protectionLevel="signature" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:allowBackup="false" >
        <!-- Service using Messenger -->
        <!-- *** POINT 2 *** Require the in-house signature perm
ission -->
        <!-- *** POINT 3 *** Do not define the intent filter and
 explicitly set the exported attribute to true. -->
        <service
            android:name="org.jssec.android.service.inhouseservi
ce.messenger.InhouseMessengerService"
            android:exported="true"
            android:permission="org.jssec.android.service.inhous
eservice.messenger.MY_PERMISSION" />
    </application>
</manifest>
```

InhouseMessengerService.java

```
package org.jssec.android.service.inhouseservice.messenger;

import org.jssec.android.shared.SigPerm;
import org.jssec.android.shared.Utils;
import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.Iterator;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.Message;
import android.os.Messenger;
import android.os.RemoteException;
import android.widget.Toast;

public class InhouseMessengerService extends Service{

    // In-house signature permission
    private static final String MY_PERMISSION = "org.jssec.andro
id.service.inhouseservice.messenger.MY_PERMISSION";
    // In-house certificate hash value
    private static String sMyCertHash = null;

    private static String myCertHash(Context context) {
        if (sMyCertHash == null) {
            if (Utils.isDebuggable(context)) {
                // Certificate hash value of debug.keystore "and
roiddebugkey"
                sMyCertHash = "0EFB7236 328348A9 89718BAD DF57F5
44 D5CCB4AE B9DB34BC 1E29DD26 F77C8255";
            } else {
                // Certificate hash value of keystore "my compan
y key"
                sMyCertHash = "D397D343 A5CBC10F 4EDDEB7C A10062
DE 5690984F 1FB9E88B D7B3A7C2 42E142CA";
            }
        }
        return sMyCertHash;
    }

    // Manage clients(destinations of sending data) in a list
    private ArrayList<Messenger> mClients = new ArrayList<Messen
ger>();
    // Messenger used when service receive data from client
    private final Messenger mMessenger = new Messenger(new Servi
ceSideHandler(mClients));
```

```
    // Handler which handles message received from client
    private static class ServiceSideHandler extends Handler{
        private ArrayList<Messenger> mClients;

        public ServiceSideHandler(ArrayList<Messenger> clients){
            mClients = clients;
        }

        @Override
        public void handleMessage(Message msg){
            switch(msg.what){
                case CommonValue.MSG_REGISTER_CLIENT:
                    // Add messenger received from client
                    mClients.add(msg.replyTo);
                    break;
                case CommonValue.MSG_UNREGISTER_CLIENT:
                    mClients.remove(msg.replyTo);
                    break;
                case CommonValue.MSG_SET_VALUE:
                    // Send data to client
                    sendMessageToClients(mClients);
                    break;
                default:
                    super.handleMessage(msg);
                    break;
            }
        }
    }

    /**
    * Send data to client
    */
    private static void sendMessageToClients(ArrayList<Messenger
> mClients){
        // *** POINT 6 *** Sensitive information can be returned
 since the requesting application is inhouse.
        String sendValue = "Sensitive information (from Service)"
;
        // Send data to the registered client one by one.
        // Use iterator to send all clients even though clients
are removed in the loop process.
        Iterator<Messenger> ite = mClients.iterator();
        while(ite.hasNext()){
            try {
                Message sendMsg = Message.obtain(null, CommonVal
ue.MSG_SET_VALUE, null);
                Bundle data = new Bundle();
                data.putString("key", sendValue);
                sendMsg.setData(data);
                Messenger next = ite.next();
                next.send(sendMsg);
            } catch (RemoteException e) {
                // If client does not exits, remove it from a li
```

```
st.
                ite.remove();
            }
        }
    }

    public IBinder onBind(Intent intent) {
        // *** POINT 4 *** Verify that the in-house signature pe
rmission is defined by an in-house application.
        if (!SigPerm.test(this, MY_PERMISSION, myCertHash(this))
) {
            Toast.makeText(this, "In-house defined signature per
mission is not defined by in-house application.", Toast.LENGTH_L
ONG).show();
            return null;
        }
        // *** POINT 5 *** Handle the received intent carefully
and securely,
        // even though the intent was sent from an in-house appl
ication.
        // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
        String param = intent.getStringExtra("PARAM");
        Toast.makeText(this, String.format("Received parameter ¥"
%s¥".", param), Toast.LENGTH_LONG).show();
        return mMessenger.getBinder();
    }

    @Override
    public void onCreate() {
        Toast.makeText(this, "Service - onCreate()", Toast.LENGT
H_SHORT).show();
    }

    @Override
    public void onDestroy() {
        Toast.makeText(this, "Service - onDestroy()", Toast.LENG
TH_SHORT).show();
    }
}
```

SigPerm.java

```
package org.jssec.android.shared;

import android.content.Context;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.PermissionInfo;

public class SigPerm {

    public static boolean test(Context ctx, String sigPermName,
String correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, sigPermName));
    }

    public static String hash(Context ctx, String sigPermName) {
        if (sigPermName == null) return null;
        try {
            // Get the package name of the application which dec
lares a permission named sigPermName.
            PackageManager pm = ctx.getPackageManager();
            PermissionInfo pi;
            pi = pm.getPermissionInfo(sigPermName, PackageManage
r.GET_META_DATA);
            String pkgname = pi.packageName;
            // Fail if the permission named sigPermName is not a
 Signature Permission
            if (pi.protectionLevel != PermissionInfo.PROTECTION_
SIGNATURE) return null;
            // Return the certificate hash value of the applicat
ion which declares a permission named sigPermName.
            return PkgCert.hash(ctx, pkgname);
        } catch (NameNotFoundException e) {
            return null;
        }
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;
```

```
public class PkgCert {

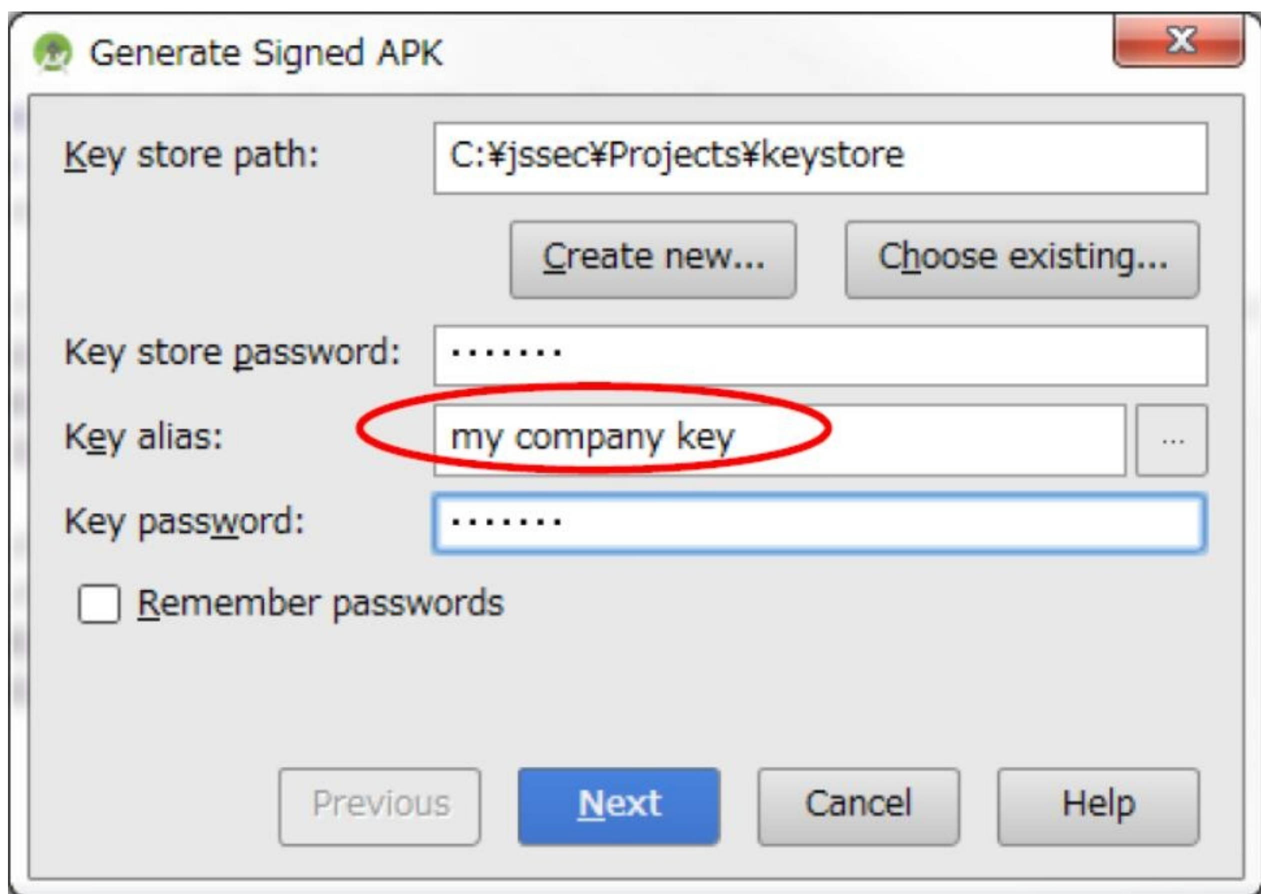
    public static boolean test(Context ctx, String pkgname, Stri
ng correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
    }
}
```

要点 7：导出 APK 时，请使用与请求应用相同的开发人员密钥对 APK 进行签名。



下面是使用内部服务的活动代码：

要点（使用服务）：

8) 声明使用内部签名权限。

9) 确认内部签名权限是由内部应用定义的。

10) 验证目标应用是否使用内部证书签名。

11) 由于目标应用是内部的，因此可以发送敏感信息。

12) 使用显式意图调用内部服务。

13) 即使数据来自内部应用，也要小心并安全地处理收到的结果数据。

14) 导出 APK 时，请使用与目标应用相同的开发人员密钥对 APK 进行签名。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.service.inhouseservice.messengeru
ser" >
    <!-- *** POINT 8 *** Declare to use the in-house signature p
ermission. -->
    <uses-permission
    android:name="org.jssec.android.service.inhouseservice.messe
nger.MY_PERMISSION" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:allowBackup="false" >
        <activity
            android:name="org.jssec.android.service.inhouseservi
ce.messengeruser.InhouseMessengerUserActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

InhouseMessengerUserActivity.java

```
package org.jssec.android.service.inhouseservice.messengeruser;

import org.jssec.android.shared.PkgCert;
import org.jssec.android.shared.SigPerm;
import org.jssec.android.shared.Utils;
import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.Message;
import android.os.Messenger;
import android.os.RemoteException;
import android.view.View;
import android.widget.Toast;
```

```
public class InhouseMessengerUserActivity extends Activity {

    private boolean mIsBound;
    private Context mContext;
    // Destination (Requested) service application information
    private static final String TARGET_PACKAGE = "org.jssec.andr
oid.service.inhouseservice.messenger";
    private static final String TARGET_CLASS = "org.jssec.androi
d.service.inhouseservice.messenger.InhouseMessengerService";
    // In-house signature permission
    private static final String MY_PERMISSION = "org.jssec.andro
id.service.inhouseservice.messenger.MY_PERMISSION";
    // In-house certificate hash value
    private static String sMyCertHash = null;

    private static String myCertHash(Context context) {
        if (sMyCertHash == null) {
            if (Utils.isDebuggable(context)) {
                // Certificate hash value of debug.keystore "and
roiddebugkey"
                sMyCertHash = "0EFB7236 328348A9 89718BAD DF57F5
44 D5CCB4AE B9DB34BC 1E29DD26 F77C8255";
            } else {
                // Certificate hash value of keystore "my compan
y key"
                sMyCertHash = "D397D343 A5CBC10F 4EDDEB7C A10062
DE 5690984F 1FB9E88B D7B3A7C2 42E142CA";
            }
        }
        return sMyCertHash;
    }

    // Messenger used when this application receives data from s
ervice.
    private Messenger mServiceMessenger = null;
    // Messenger used when this application sends data to servic
e.
    private final Messenger mActivityMessenger = new Messenger(n
ew ActivitySideHandler());

    // Handler which handles message received from service
    private class ActivitySideHandler extends Handler {

        @Override
        public void handleMessage(Message msg) {
            switch (msg.what) {
                case CommonValue.MSG_SET_VALUE:
                    Bundle data = msg.getData();
                    String info = data.getString("key");
                    // *** POINT 13 *** Handle the received resu
lt data carefully and securely,
                    // even though the data came from an in-hous
e application
```

```
                    // Omitted, since this is a sample. Please r
efer to "3.2 Handling Input Data Carefully and Securely."
                    Toast.makeText(mContext, String.format("Rece
ived ¥"%s¥" from service.", info),
                    Toast.LENGTH_SHORT).show();
                    break;
                default:
                    super.handleMessage(msg);
            }
        }

    }

    // Connection used to connect with service. This is necessar
y when service is implemented with bindSrvice().
    private ServiceConnection mConnection = new ServiceConnectio
n() {

        // This is called when the connection with the service h
as been established.
        @Override
        public void onServiceConnected(ComponentName className, 
IBinder service) {
            mServiceMessenger = new Messenger(service);
            Toast.makeText(mContext, "Connect to service", Toast
.LENGTH_SHORT).show();
            try {
                // Send own messenger to service
                Message msg = Message.obtain(null, CommonValue.M
SG_REGISTER_CLIENT);
                msg.replyTo = mActivityMessenger;
                mServiceMessenger.send(msg);
            } catch (RemoteException e) {
                // Service stopped abnormally
            }
        }

        // This is called when the service stopped abnormally an
d connection is disconnected.
        @Override
        public void onServiceDisconnected(ComponentName classNam
e) {
            mServiceMessenger = null;
            Toast.makeText(mContext, "Disconnected from service"
, Toast.LENGTH_SHORT).show();
        }
    };

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.inhouseservice_activity);
        mContext = this;
```

```
    }

    // --- StartService control ---
    public void onStartServiceClick(View v) {
        // Start bindService
        doBindService();
    }

    public void onGetInfoClick(View v) {
        getServiceinfo();
    }

    public void onStopServiceClick(View v) {
        doUnbindService();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        doUnbindService();
    }

    /**
    * Connect to service
    */
    void doBindService() {
        if (!mIsBound){
            // *** POINT 9 *** Verify that the in-house signatur
e permission is defined by an in-house application.
            if (!SigPerm.test(this, MY_PERMISSION, myCertHash(th
is))) {
                Toast.makeText(this, "In-house defined signature
 permission is not defined by in-house application.", Toast.LENG
TH_LONG).show();
                return;
            }
            // *** POINT 10 *** Verify that the destination appl
ication is signed with the in-house certificate.
            if (!PkgCert.test(this, TARGET_PACKAGE, myCertHash(t
his))) {
                Toast.makeText(this, "Destination(Requested) ser
vice application is not in-house application.", Toast.LENGTH_LON
G).show();
                return;
            }
            Intent intent = new Intent();
            // *** POINT 11 *** Sensitive information can be sen
t since the destination application is in-house one.
            intent.putExtra("PARAM", "Sensitive information");
            // *** POINT 12 *** Use the explicit intent to call
an in-house service.
            intent.setClassName(TARGET_PACKAGE, TARGET_CLASS);
            bindService(intent, mConnection, Context.BIND_AUTO_C
```

```
REATE);
            mIsBound = true;
        }
    }

    /**
    * Disconnect service
    */
    void doUnbindService() {
        if (mIsBound) {
            unbindService(mConnection);
            mIsBound = false;
        }
    }

    /**
    * Get information from service
    */
    void getServiceinfo() {
        if (mServiceMessenger != null) {
            try {
                // Request sending information
                Message msg = Message.obtain(null, CommonValue.M
SG_SET_VALUE);
                mServiceMessenger.send(msg);
            } catch (RemoteException e) {
                // Service stopped abnormally
            }
        }
    }
}
```

SigPerm.java

```
package org.jssec.android.shared;

import android.content.Context;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.PermissionInfo;

public class SigPerm {

    public static boolean test(Context ctx, String sigPermName,
String correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, sigPermName));
    }

    public static String hash(Context ctx, String sigPermName) {
        if (sigPermName == null) return null;
        try {
            // Get the package name of the application which dec
lares a permission named sigPermName.
            PackageManager pm = ctx.getPackageManager();
            PermissionInfo pi;
            pi = pm.getPermissionInfo(sigPermName, PackageManage
r.GET_META_DATA);
            String pkgname = pi.packageName;
            // Fail if the permission named sigPermName is not a
 Signature Permission
            if (pi.protectionLevel != PermissionInfo.PROTECTION_
SIGNATURE) return null;
            // Return the certificate hash value of the applicat
ion which declares a permission named sigPermName.
            return PkgCert.hash(ctx, pkgname);
        } catch (NameNotFoundException e) {
            return null;
        }
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;
```

```
public class PkgCert {

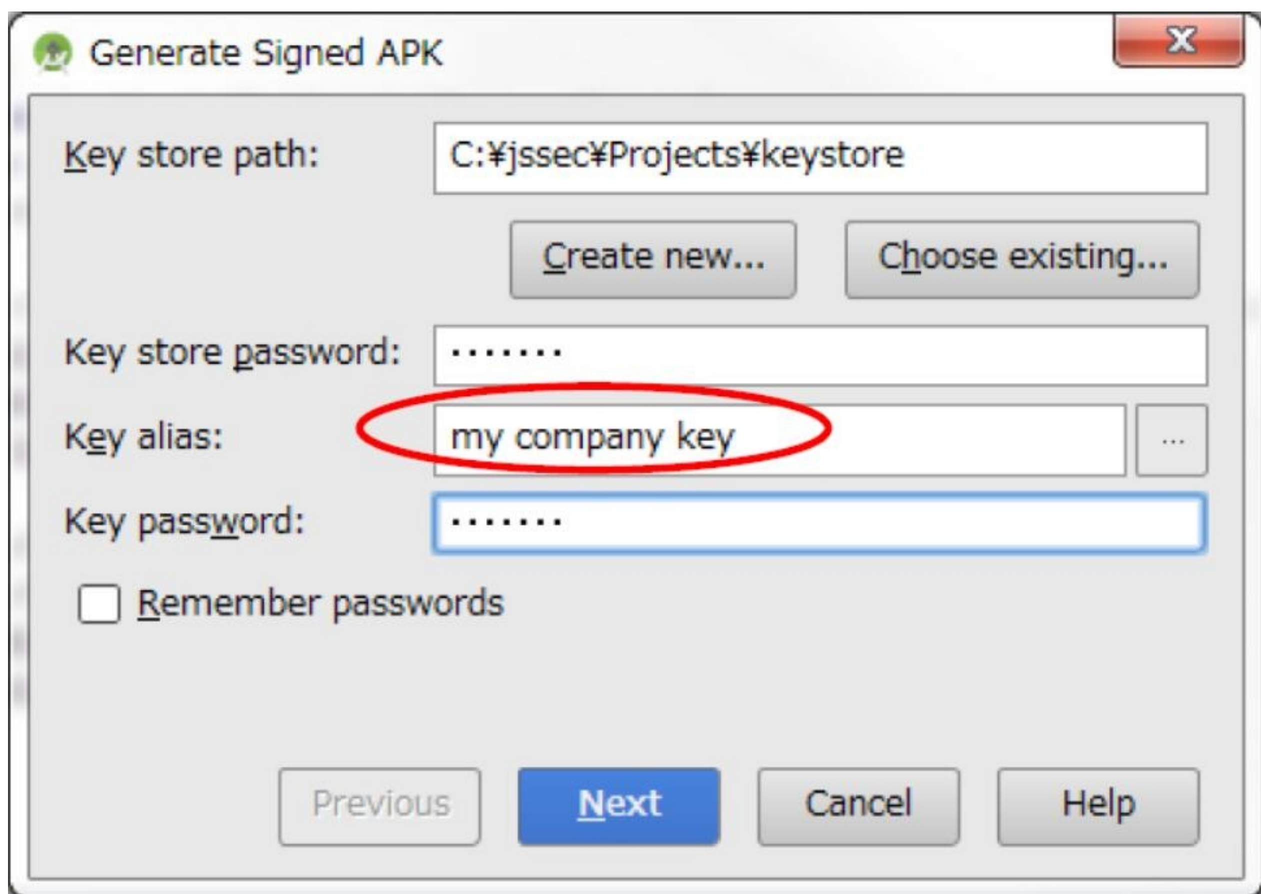
    public static boolean test(Context ctx, String pkgname, String correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, PackageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; // Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(data);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
    }
}
```

要点 14：导出 APK 时，请使用与目标应用相同的开发人员密钥对 APK 进行签名。


Generate Signed APK
Key store path:
C:¥jssec¥Projects¥keystore
Create new...
Choose existing...
Key store password:
Key alias:
my company key
Key password:
Remember passwords
Previous
Next
Cancel
Help

4.4.2 规则书

实现或使用服务时，遵循下列规则。

4.4.2.1 仅仅在应用中使用的服务，必须设为私有（必需）

仅在应用（或同一个 UID）中使用的服务必须设置为“私有”。 它避免了应用意外地从其他应用接收意图，并最终防止应用的功能被使用，或应用的行为变得异常。

在 `AndroidManifest.xml` 中定义服务时，你在必须将导出属性设置为 `false` 。

AndroidManifest.xml

```
<!-- Private Service derived from Service class -->
<!-- *** POINT 1 *** Set false for the exported attribute explic
itly. -->
<service android:name=".PrivateStartService" android:exported="f
alse"/>
```

另外，这种情况很少见，但是当服务仅在应用中使用时，不要设置意图过滤器。原因是，由于意图过滤器的特性，可能会意外调用其他应用中的公共服务，虽然你打算调用应用内的私有服务。

AndroidManifest.xml（不推荐）

```
<!-- Private Service derived from Service class -->
<!-- *** POINT 1 *** Set false for the exported attribute explic
itly. -->
<service android:name=".PrivateStartService" android:exported="f
alse">
    <intent-filter>
        <action android:name="org.jssec.android.service.OPEN />
    </intent-filter>
</service>
```

请参阅“4.4.3.1 导出属性和意图过滤器设置的组合（在服务情况下）”。

4.4.2.2 小心并安全地处理收到的数据（必需）

与“活动”相同，如果是“服务”，则在处理收到的意图数据时，你应该做的第一件事是输入验证。 同样在服务的用户方，有必要验证来自服务的结果信息的安全性。请参阅“4.1.2.5 小心并安全地处理收到的意图（必需）”和“4.1.2.9 小心并安全地处理从被请求活动返回的数据”。

在服务中，你还应该小心实现调用方法，并通过消息交换数据。

请参阅“3.2 小心并安全地处理输入数据”。

4.4.2.3 在验证签名权限由内部定义之后，使用内部定义的签名全新啊（必需）

确保在创建服务时，通过定义内部签名权限来保护你的内部服务。 由于在 `AndroidManifest.xml` 文件中定义权限或声明权限请求，没有提供足够的安全性，请务必参考“5.2.1.2 如何使用内部定义的签名权限在内部应用之间进行通信”。

4.4.2.4 不要在 `onCreate` 中判断服务是否提供自己的函数（必需）

`onCreate` 中不应包含安全检查，例如意图参数验证，或内部定义的签名权限验证，因为在服务运行期间接收到新请求时，不会执行 `onCreate` 过程。 所以，在实现由 `startService` 启动的服务时，应该由 `onStartCommand` 来执行判断（使用 `IntentService` 的情况下，判断应该由 `onHandleIntent` 来执行）。在实现由 `bindService` 启动的 `Service` 的情况下也是一样的，判断应该由 `onBind` 执行。

4.4.2.5 返回结果信息，注意来自目标应用的可能的信息泄露（必需）

取决于服务类型，结果信息的目标应用（回调接收方/ `Message` 的目标）的可靠性有所不同。 考虑到目标可能是恶意软件的可能性，需要认真考虑信息泄漏。

详细信息请参阅“4.1.2.7 返回结果时，注意目标应用的可能的信息泄露（必需）”。

4.4.2.6 如果目标是固定的，使用显式意图（必需）

当通过隐式意图使用服务时，如果意图过滤器的定义相同，则意图会发送到首先之前的服务。 如果之前安装了恶意软件，它故意定义了同一个意图过滤器，则意图会发送到恶意软件并发生信息泄露。 另一方面，当通过显式意图使用服务时，只有预期的服务会收到意图，所以这样更安全。

还有一些要考虑的要点，请参阅“4.1.2.8 如果目标活动是预定义的，则使用显式意图（必需）”。

4.4.2.7 如果与其他公司的应用链接，验证目标服务（必需）

与其他公司的应用链接时，确保确定了白名单。 你可以通过在应用内保存公司证书的散列副本，并使用目标应用的证书散列来检查它。 这将防止恶意应用伪造意图。 具体实现方法请参考“4.4.1.3 创建/使用伙伴服务”的示例代码部分。

4.4.2.8 当提供二次素材时，素材应该受到相同级别的保护（必需）

当受到权限保护的信息或功能素材，由另一个应用提供时，你需要确保它具有访问素材所需的相同权限。 在 Android OS 权限安全模型中，只有已被授予适当权限的应用，才能直接访问受保护的素材。 但是，存在一个漏洞，因为具有素材权限的应用可以充当代理，并允许非特权应用访问。 基本上这与重新授权相同，因此它被称为“重新授权”问题。 请参阅“5.2.3.4 重新授权问题”。

4.4.2.9 尽可能不要发送敏感信息（推荐）

你不应将敏感信息发送给不受信任的各方。

在与服务交换敏感信息时，你需要考虑信息泄露的风险。 你必须假设，发送到公共服务的意图中的所有数据都可以由恶意第三方获取。 此外，根据实现情况，向伙伴或内部服务发送意图时，也存在各种信息泄露的风险。

首先，不发送敏感数据，是防止信息泄露的唯一完美解决方案，因此你应该尽可能限制发送的敏感信息的数量。 当需要发送敏感信息时，最佳做法是仅发送给可信服务并确保信息不会通过 `LogCat` 泄漏。

4.4.3 高级话题

4.4.3.1 导出属性和意图过滤器设置的组合（在服务情况下）

我们已经本指南中解释了如何在实现四种服务类型：私有服务，公共服务，伙伴服务和内部服务。 下表中定义了每种导出属性类型的许可设置，以及 `intent-filter` 元素的各种组合，它们 `AndroidManifest.xml` 文件中定义。 请验证导出属性和 `intent-filter` 元素与你尝试创建的服务的兼容性。

表 4.4-3

	导出属性的值		
	True	False	未指定
意图过滤器已定义	公共	（不使用）	（不使用）
意图过滤器未定义	公共，伙伴，内部	私有	（不使用）

如果服务中的导出属性是未指定的，服务是否公开由是否定义了意图过滤器决定 [9]；但是，在本指南中，禁止将服务的导出属性设置为未指定。 通常，如前所述，最好避免依赖任何给定 API 的默认行为的实现; 此外，如果存在显式方法来配置重要的安全相关设置，例如导出属性，那么使用这些方法总是一个好主意。

> [9] 如果定义了任何意图过滤器，服务是公开的，否则是私有的。更多信息请见 https://developer.android.com/guide/topics/manifest/service-element.html#exported 。

不应该使用未定义的意图过滤器和导出属性 `false` 的原因是，Android 的行为存在漏洞，并且由于意图过滤器的工作原理，可能会意外调用其他应用的服务。

具体而言，Android 的行为如下，因此在设计应用时需要仔细考虑。

- 当多个服务定义了相同的意图过滤器内容时，更早安装的应用中的服务是优先的。
- 如果使用显式意图，则优先的服务将被自动选择并由 OS 调用。

以下三张图描述了一个系统，由于 Android 行为而发生意外调用的。 图 4.4-4 是一个正常行为的例子，私有服务（应用 A）只能由同一个应用通过隐式意图调用。 因为只有应用 A 定义了意图过滤器（图中的 `action ="X"` ），所以它的行为正常。 这是正常的行为。

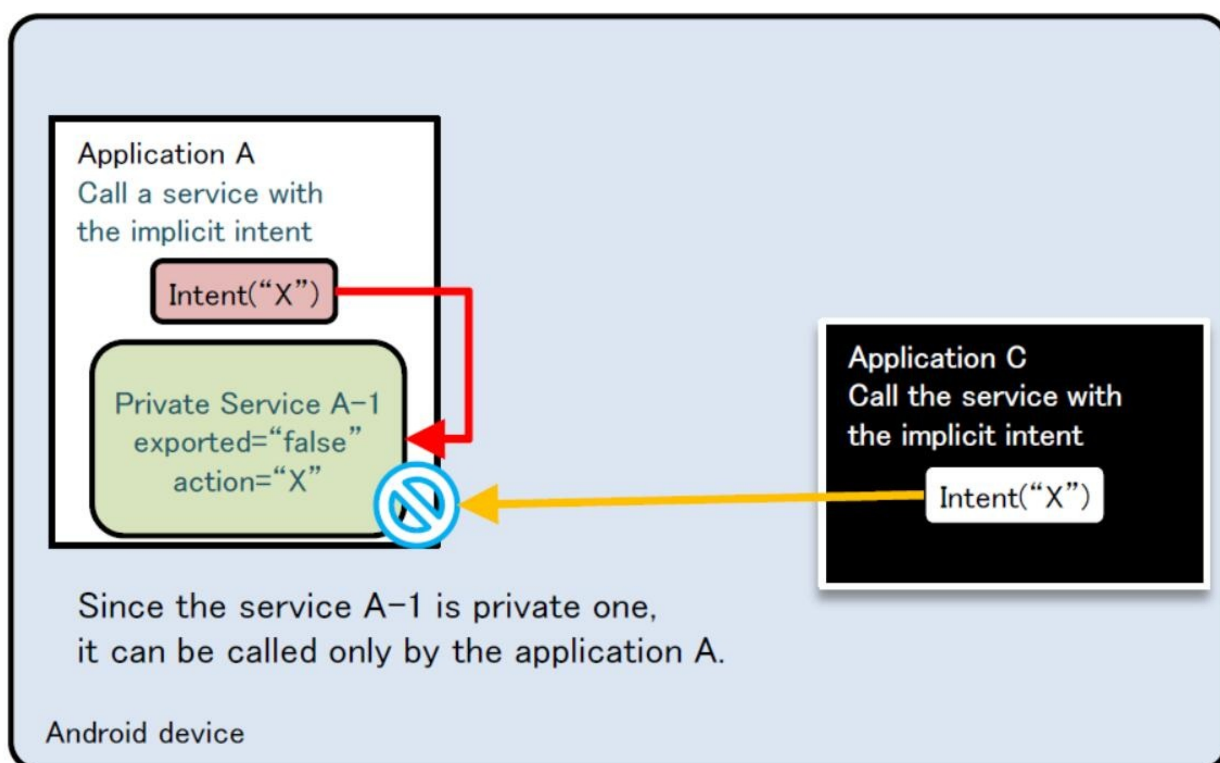


Figure 4.4-4

图 4.4-5 和图 4.4-6 展示了一个情景，其中应用 B 和应用 A 中定义了相同的意图过滤器（ `action ="X"` ）。

图 4.4-5 展示了应用按 `A -> B` 的顺序安装。在这种情况下，当应用 C 发送隐式意图时，私有服务（A-1）调用失败。 另一方面，由于应用 A 可以通过隐式意图，按照预期成功调用应用内的私有服务，因此在安全性（恶意软件的对策）方面不会有任何问题。

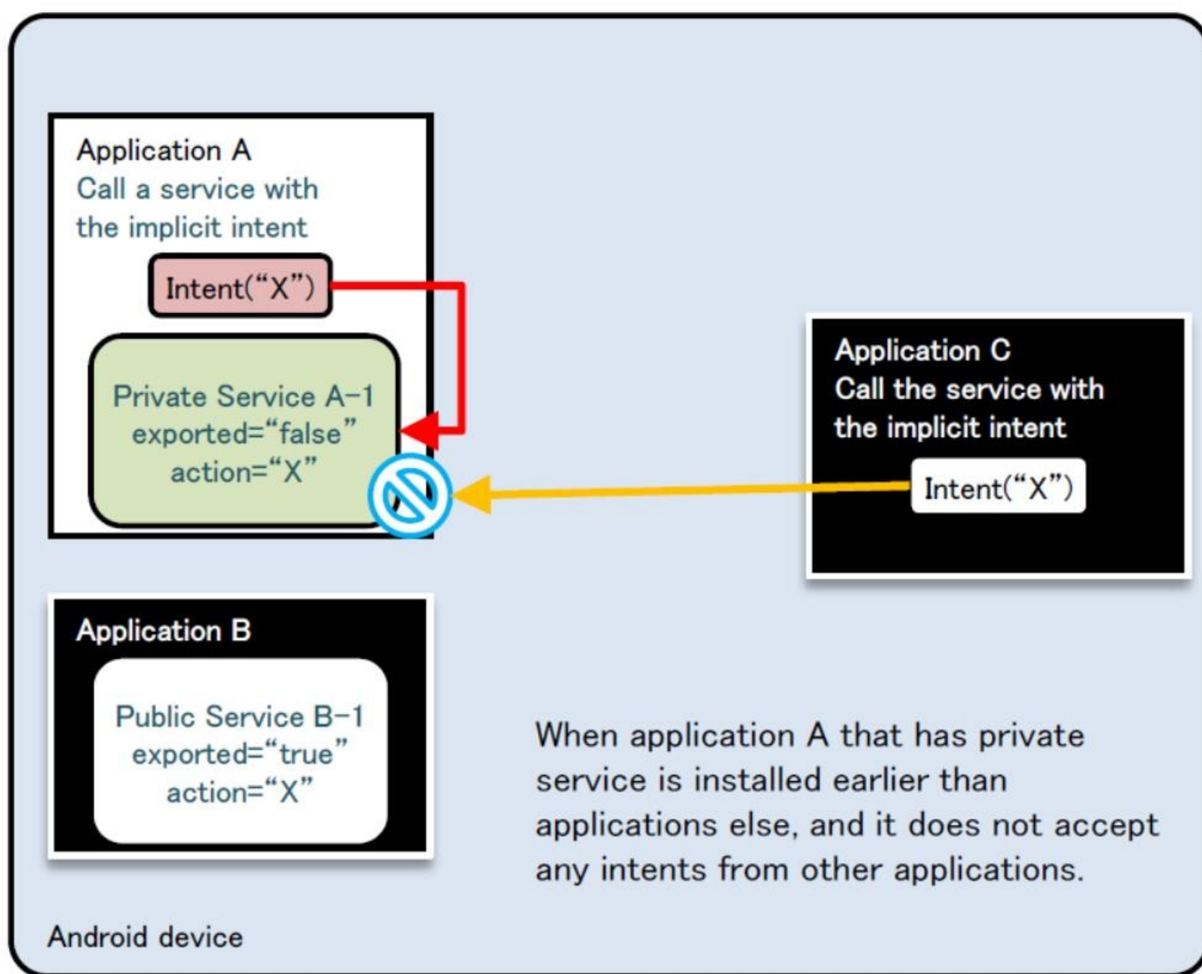


Figure 4.4-5

图 4.4-6 展示了一个场景，应用以 `B->A` 的顺序安装。 就安全性而言，这里存在一个问题，应用 A 尝试通过发送隐式意图来，调用应用中的私有服务，但实际上调用了之前安装的应用 B 中的公共活动（B-1）。 由于这个漏洞，敏感信息可能会从应用 A 发送到应用 B。 如果应用 B 是恶意软件，它会导致敏感信息的泄漏。

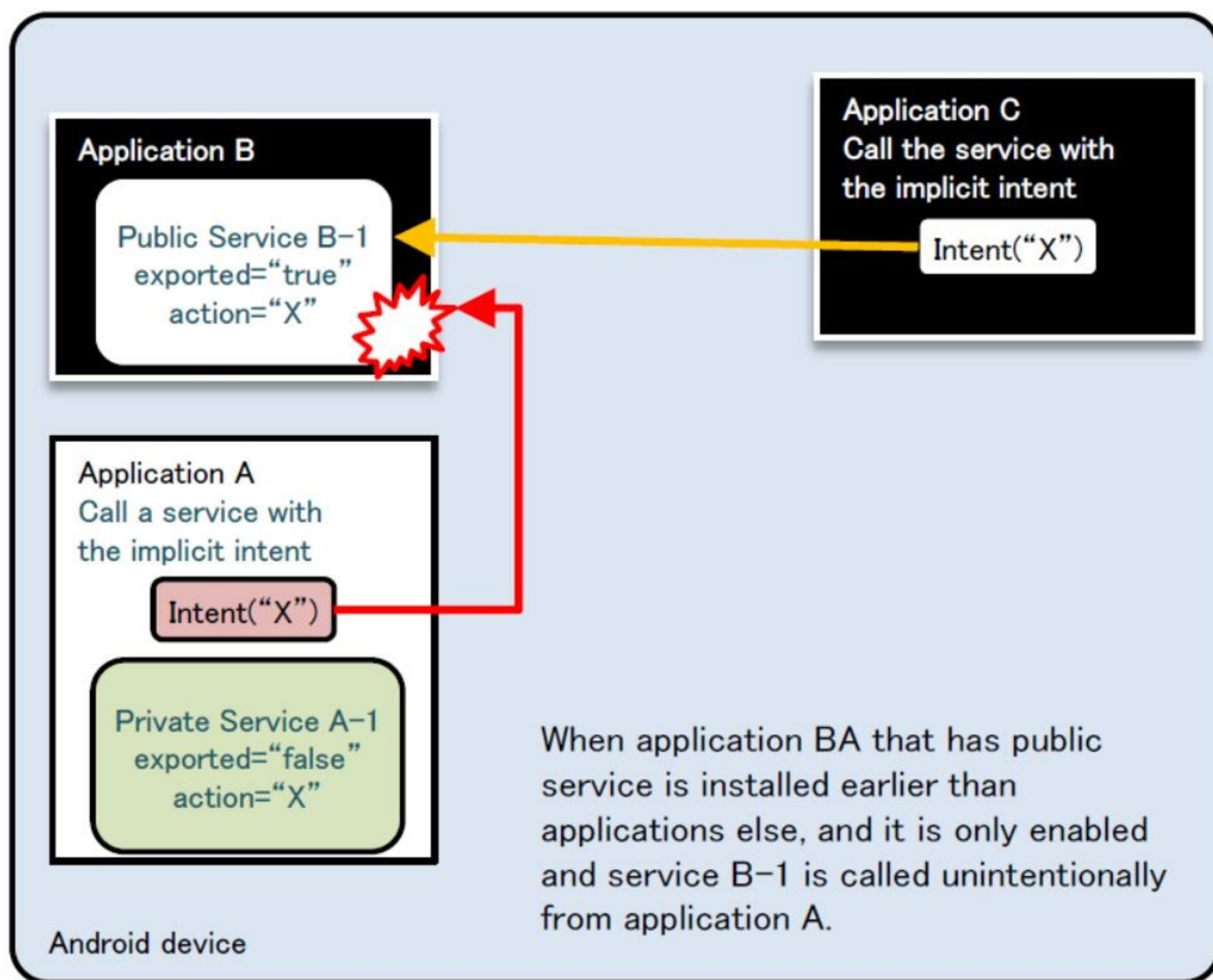


Figure 4.4-6

如上所示，使用意图过滤器向私有服务发送隐式意图，可能会导致意外行为，因此最好避免此设置。

4.4.3.2 如何实现服务

由于实现服务的方法是多种多样的，应该按安全类型进行选择，它由示例代码分类，本文对各个特性进行了简要说明。 它大致分为使用 `startService` 和使用 `bindService` 的情况。 还可以创建在 `startService` 和 `bindService` 中都可以使用的服务。 应该调查以下项目来确定服务的实现方法。

- 是否将服务公开给其他应用（服务的公开）
- 是否在运行中交换数据（相互发送/接收数据）
- 是否控制服务（启动或完成）
- 是否作为另一个进程执行（进程间通信）
- 是否并行执行多个进程（并行进程）

表 4.4-3 显示了每个条目的实现方法类别和可行性。 “NG”代表不可能的情况，或者需要另一个框架的情况，它与所提供的函数不同。

表 4.4-4 服务的实现方法分类

类别	服务公开	相互发送/接收数据	控制服务	进程间通信	并行进程
`startService` 类型	OK	NG	OK	OK	NG
`IntentService` 类型	OK	NG	NG	OK	NG
本地绑定类型	NG	OK	OK	NG	NG
`Messenger` 绑定类型	OK	OK	OK	OK	NG
AIDL 绑定类型	OK	OK	OK	OK	OK

`startService` 类型

这是最基本的服务。 它继承了 `Service` 类，并通过 `onStartCommand` 执行过程。

在用户方，服务由意图指定，并通过 `startService` 调用。 由于结果等数据无法直接返回给源意图，因此应与其他方法（如广播）结合使用。 具体示例请参考“4.4.1.1 创建/使用私有服务”。

安全性检查应该由 `onStartCommand` 完成，但不能用于伙伴服务，因为无法获取来源的软件包名称。

`IntentService` 类型

`IntentService` 是通过继承 `Service` 创建的类。 调用方法与 `startService` 类型相同。 以下是与标准服务（ `startService` 类型）相比较的特征。

- 意图的处理由 `onHandleIntent` 完成（不使用 `onStartCommand` ）。
- 由另一个线程执行。
- 过程将排队。

由于过程是由另一个线程执行的，因此调用会立即返回，并且面向意图的过程由队列系统顺序执行。 每个意图并不是并行处理的，但根据产品的要求，它也可以作为选项来选择，来简化实现。由于结果等数据不能返回给源意图，因此应该与其他方法（如广播）结合使用。 具体实例请参考“4.4.1.2 创建/使用公共服务”。

安全性检查应该由 `onHandleIntent` 来完成，但不能用于伙伴服务，因为无法获取来源的包名称。

本地绑定类型

这是一种实现本地服务的方法，它仅工作在与应用相同的过程中。 将类定义为从 `Binder` 类派生的类，并准备将 `Service` 中实现的特性（方法）提供给调用方。

在用户方，服务由意图指定并使用 `bindService` 调用。 这是绑定服务的所有方法中最简单的实现，但它的用途有限，因为它不能被其他进程启动，并且服务也不能公开。 具体实现示例，请参阅示例代码中包含的项目“ `PrivateServiceLocalBind` 服务”。

从安全角度来看，只能实现私有服务。

`Messenger` 绑定类型

这是一种方法，通过使用 `Messenger` 系统来实现与服务的链接。

由于 `Messenger` 可以提供为来自服务用户方的 `Message` 目标，因此可以相对容易地实现数据交换。 另外，由于过程要进行排队，因此它具有“线程安全”的特性。 每个过程不可能并行，但根据产品的要求，它也可以作为选项来选择，来简化实现。 在用户端，服务由意图指定，通过 `bindService` 调用，具体实现示例请参见“4.4.1.4 创建/使用内部服务”。

安全检查需要在 `onBind` 或 `Message Handler` 中进行，但不能 用于伙伴服务，因为无法获取来源的包名称。

AIDL 绑定类型

这是一种方法，通过使用 AIDL 系统实现与服务的链接。 接口通过 AIDL 定义，并将服务拥有的特性提供为方法。 另外，回调也可以通过在用户端实现由 AIDL 定义的接口来实现，多线程调用是可能的，但有必要在服务端明确实现互斥。

用户端可以通过指定意图并使用 `bindService` 来调用服务。 具体实现示例请参考“4.4.1.3 创建/使用伙伴服务”。

安全性检查必须在 `onBind` 中为内部服务执行，以及由 AIDL 为伙伴服务定义的接口的每种方法执行。

这可以用于本指南中描述的所有安全类型的服务。

4.5 使用 SQLite

通过使用 SQLite 创建/操作数据库时，在安全性方面有一些警告。 要点是合理设置数据库文件的访问权限，以及 SQL 注入的对策。 允许从外部直接读取/写入数据库文件（在多个应用程序之间共享）的数据库不在此处，假设在内容供应器的后端和应用本身中使用该数据库。 另外，在处理不太多敏感信息的情况下，建议采取下述对策，尽管这里可以处理一定程度的敏感信息。

4.5.1 示例代码

4.5.1.1 创建/操作数据库

在 Android 应用中处理数据库时，可以通过使用 `SQLiteOpenHelper` [10] 来实现数据库文件的适当安排和访问权限设置（拒绝其他应用访问的设置）。 下面是一个简单的应用示例，它在启动时创建数据库，并通过 UI 执行搜索/添加/更改/删除数据。 示例代码完成了 SQL 注入的防范，来避免来自外部的输入执行不正确的 SQL。

> [10] 对于文件存储，可以将绝对文件路径指定为 `SQLiteOpenHelper` 构造函数的第二个参数（名称）。 因此，如果指定了 SD 卡路径，则需要注意，存储的文件可以被其他应用读取和写入。

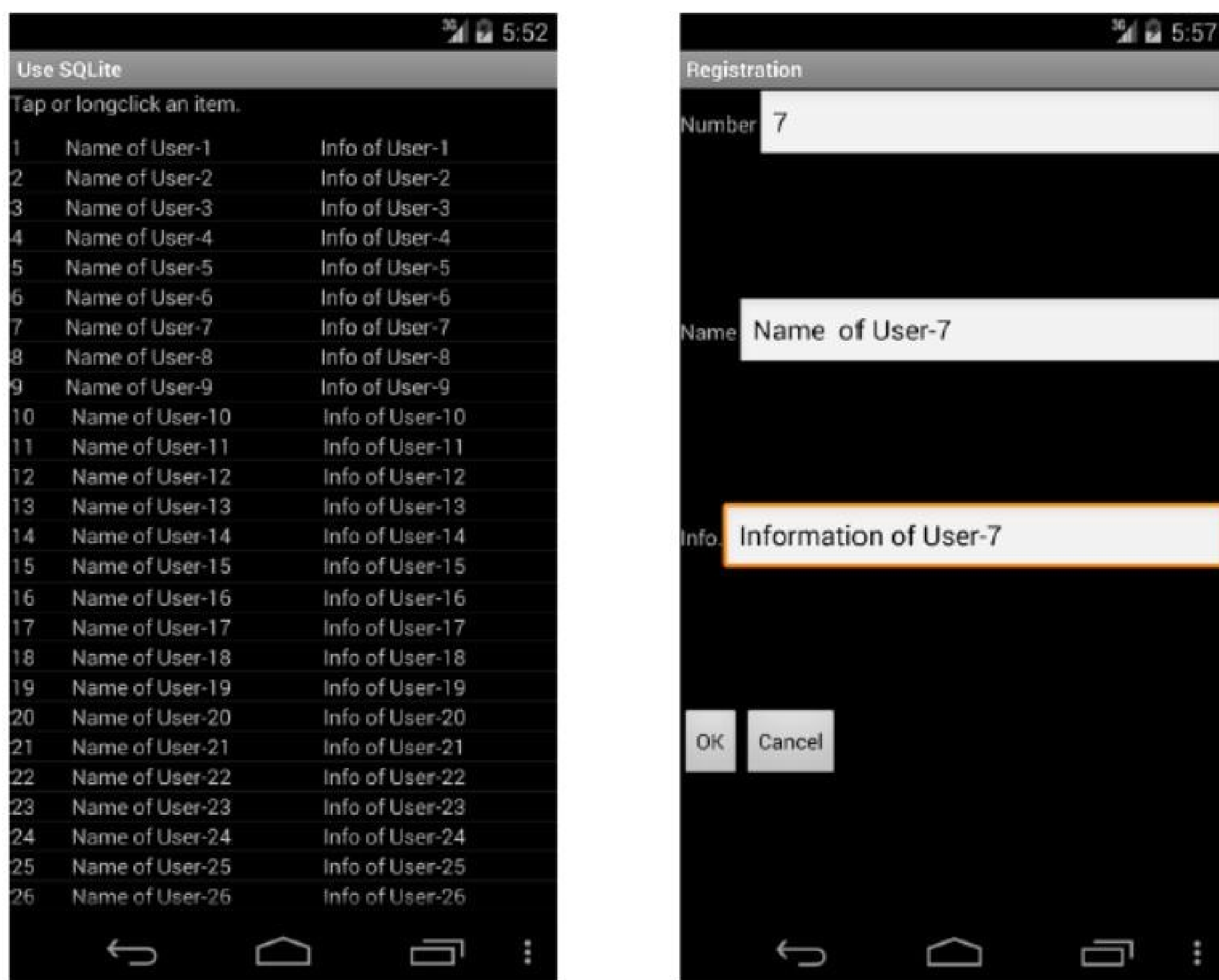


Figure 4.5-1

1) `SQLiteOpenHelper` 应该用于创建数据库。

2) 使用占位符。

3) 根据应用要求验证输入值。

SampleDbOpenHelper.java

```
package org.jssec.android.sqlite;

import android.content.Context;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
import android.widget.Toast;

public class SampleDbOpenHelper extends SQLiteOpenHelper {

    private SQLiteDatabase mSampleDb; //Database to store the da
ta to be handled

    public static SampleDbOpenHelper newHelper(Context context)
{
        //*** POINT 1 *** SQLiteOpenHelper should be used for da
tabase creation.
        return new SampleDbOpenHelper(context);
    }

    public SQLiteDatabase getDb() {
        return mSampleDb;
    }

    //Open DB by Writable mode
    public void openDatabaseWithHelper() {
        try {
            if (mSampleDb != null && mSampleDb.isOpen()) {
                if (!mSampleDb.isReadOnly())// Already opened by
 writable mode
                    return;
                mSampleDb.close();
            }
            mSampleDb = getWritableDatabase(); //It's opened her
e.
        } catch (SQLException e) {
            //In case fail to construct database, output to log
            Log.e(mContext.getClass().toString(), mContext.getSt
ring(R.string.DATABASE_OPEN_ERROR_MESSAGE));
            Toast.makeText(mContext, R.string.DATABASE_OPEN_ERRO
R_MESSAGE, Toast.LENGTH_LONG).show();
        }
    }

    //Open DB by ReadOnly mode.
    public void openDatabaseReadOnly() {
        try {
            if (mSampleDb != null && mSampleDb.isOpen()) {
                if (mSampleDb.isReadOnly())// Already opened by
ReadOnly.
                    return;
```

```
                mSampleDb.close();
            }
            SQLiteDatabase.openDatabase(mContext.getDatabasePath
(CommonData.DBFILE_NAME).getPath(), null, SQLiteDatabase.OPEN_RE
ADONLY);
        } catch (SQLException e) {
            //In case failed to construct database, output to log

            Log.e(mContext.getClass().toString(), mContext.getSt
ring(R.string.DATABASE_OPEN_ERROR_MESSAGE));
            Toast.makeText(mContext, R.string.DATABASE_OPEN_ERRO
R_MESSAGE, Toast.LENGTH_LONG).show();
        }
    }

    //Database Close
    public void closeDatabase() {
        try {
            if (mSampleDb != null && mSampleDb.isOpen()) {
                mSampleDb.close();
            }
        } catch (SQLException e) {
            //In case failed to construct database, output to log

            Log.e(mContext.getClass().toString(), mContext.getSt
ring(R.string.DATABASE_CLOSE_ERROR_MESSAGE));
            Toast.makeText(mContext, R.string.DATABASE_CLOSE_ERR
OR_MESSAGE, Toast.LENGTH_LONG).show();
        }
    }

    //Remember Context
    private Context mContext;
    //Table creation command
    private static final String CREATE_TABLE_COMMANDS
        = "CREATE TABLE " + CommonData.TABLE_NAME + " ("
        + "_id INTEGER PRIMARY KEY AUTOINCREMENT, "
        + "idno INTEGER UNIQUE, "
        + "name VARCHAR(" + CommonData.TEXT_DATA_LENGTH_MAX + ")
 NOT NULL, "
        + "info VARCHAR(" + CommonData.TEXT_DATA_LENGTH_MAX + ")"

        + ");";

    public SampleDbOpenHelper(Context context) {
        super(context, CommonData.DBFILE_NAME, null, CommonData.
DB_VERSION);
        mContext = context;
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        try {
```

```
            db.execSQL(CREATE_TABLE_COMMANDS); //Execute DB cons
truction command
        } catch (SQLException e) {
            //In case failed to construct database, output to log

            Log.e(this.getClass().toString(), mContext.getString
(R.string.DATABASE_CREATE_ERROR_MESSAGE));
        }
    }

    @Override
    public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg
2) {
        // It's to be executed when database version up. Write p
rocesses like data transition.
    }
}
```

DataSearchTask.java（SQLite 数据库项目）

```
package org.jssec.android.sqlite.task;

import org.jssec.android.sqlite.CommonData;
import org.jssec.android.sqlite.DataValidator;
import org.jssec.android.sqlite.MainActivity;
import org.jssec.android.sqlite.R;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.os.AsyncTask;
import android.util.Log;

//Data search task
public class DataSearchTask extends AsyncTask<String, Void, Curs
or> {

    private MainActivity mActivity;
    private SQLiteDatabase mSampleDB;

    public DataSearchTask(SQLiteDatabase db, MainActivity activi
ty) {
        mSampleDB = db;
        mActivity = activity;
    }

    @Override
    protected Cursor doInBackground(String... params) {
        String idno = params[0];
        String name = params[1];
        String info = params[2];
```

```
        String cols[] = {"_id", "idno","name","info"};
        Cursor cur;
        //*** POINT 3 *** Validate the input value according the
 application requirements.
        if (!DataValidator.validateData(idno, name, info)){
            return null;
        }
        //When all parameters are null, execute all search
        if ((idno == null || idno.length() == 0) &&
            (name == null || name.length() == 0) &&
            (info == null || info.length() == 0) ) {
            try {
                cur = mSampleDB.query(CommonData.TABLE_NAME, col
s, null, null, null, null, null);
            } catch (SQLException e) {
                Log.e(DataSearchTask.class.toString(), mActivity
.getString(R.string.SEARCHING_ERROR_MESSAGE));
                return null;
            }
            return cur;
        }
        //When No is specified, execute searching by No
        if (idno != null && idno.length() > 0) {
            String selectionArgs[] = {idno};
            try {
                //*** POINT 2 *** Use place holder.
                cur = mSampleDB.query(CommonData.TABLE_NAME, col
s, "idno = ?", selectionArgs, null, null, null);
            } catch (SQLException e) {
                Log.e(DataSearchTask.class.toString(), mActivity
.getString(R.string.SEARCHING_ERROR_MESSAGE));
                return null;
            }
            return cur;
        }
        //When Name is specified, execute perfect match search b
y Name
        if (name != null && name.length() > 0) {
            String selectionArgs[] = {name};
            try {
                //*** POINT 2 *** Use place holder.
                cur = mSampleDB.query(CommonData.TABLE_NAME, col
s, "name = ?", selectionArgs, null, null, null);
            } catch (SQLException e) {
                Log.e(DataSearchTask.class.toString(), mActivity
.getString(R.string.SEARCHING_ERROR_MESSAGE));
                return null;
            }
            return cur;
        }
        //Other than above, execute partly match searching with
the condition of info.
        String argString = info.replaceAll("@", "@@"); //Escape
```

```
$ in info which was received as input.
        argString = argString.replaceAll("%", "@%"); //Escape %
in info which was received as input.
        argString = argString.replaceAll("_", "@_"); //Escape _
in info which was received as input.
        String selectionArgs[] = {argString};
        try {
            //*** POINT 2 *** Use place holder.
            cur = mSampleDB.query(CommonData.TABLE_NAME, cols, "
info LIKE '%' || ? || '%' ESCAPE '@'", selectionArgs, null, null
, null);
        } catch (SQLException e) {
            Log.e(DataSearchTask.class.toString(), mActivity.get
String(R.string.SEARCHING_ERROR_MESSAGE));
            return null;
        }
        return cur;
    }

    @Override
    protected void onPostExecute(Cursor resultCur) {
        mActivity.updateCursor(resultCur);
    }
}
```

DataValidator.java

```
package org.jssec.android.sqlite;

public class DataValidator {

    //Validate the Input value
    //validate numeric characters
    public static boolean validateNo(String idno) {
        //null and blank are OK
        if (idno == null || idno.length() == 0) {
            return true;
        }
        //Validate that it's numeric character.
        try {
                    if (!idno.matches("[1-9][0-9]*")) {
                //Error if it's not numeric value
                return false;
            }
        } catch (NullPointerException e) {
            //Detected an error
            return false;
        }
        return true;
    }
```

```
    // Validate the length of a character string
    public static boolean validateLength(String str, int max_len
gth) {
        //null and blank are OK
        if (str == null || str.length() == 0) {
            return true;
        }
        //Validate the length of a character string is less than
 MAX
        try {
            if (str.length() > max_length) {
                //When it's longer than MAX, error
                return false;
            }
        } catch (NullPointerException e) {
            //Bug
            return false;
        }
        return true;
    }

    // Validate the Input value
    public static boolean validateData(String idno, String name,
 String info) {
        if (!validateNo(idno)) {
            return false;
        }
        if (!validateLength(name, CommonData.TEXT_DATA_LENGTH_MA
X)) {
            return false;
        }else if(!validateLength(info, CommonData.TEXT_DATA_LENG
TH_MAX)) {
            return false;
        }
        return true;
    }
}
```

4.5.2 规则书

使用 SQLite 时，遵循以下规则：

4.5.2.1 正确设置 DB 文件位置和访问权限（必需）

考虑到 DB 文件数据的保护，DB 文件位置和访问权限设置是需要一起考虑的非常重要的因素。 例如，即使正确设置了文件访问权，如果 DB 文件位于无法设置访问权的位置，则任何人可以访问 DB 文件，例如， SD 卡。 如果它位于应用目录中，如果访问权限设置不正确，它最终将允许意外访问。 以下是正确分配和访问权限设置的一些要点，以及实现它们的方法。 为了保护数据库文件（数据），对于位置和访问权限设置，需要执行以下两点。

1) 位置

位于可以由 `Context#getDatabasePath(String name)` 获取的文件路径，或者在某些情况下，可以由 `Context#getFilesDir11` 获取的目录。

2) 访问权限

设置为 `MODE_PRIVATE` （只能由创建文件的应用访问）模式。

通过执行以下2点，即可 创建其他应用无法访问的 DB 文件。 以下是执行它们的一些方法。

1. 使用 `SQLiteOpenHelper`
2. 使用 `Context#openOrCreateDatabase`

创建 DB 文件时，可以使用 `SQLiteDatabase#openOrCreateDatabase` 。 但是，使用此方法时，可以在某些 Android 智能手机设备中创建可从其他应用读取的 DB 文件。 所以建议避免这种方法，并使用其他方法。 上述量种方法的每个特征如下 [11]

> [11] 这两种方法都提供了（包）目录下的路径，只能由指定的应用读取和写入。

使用 `SQLiteOpenHelper`

当使用 `SQLiteOpenHelper` 时，开发人员不需要担心很多事情。 创建一个从 `SQLiteOpenHelper` 派生的类，并为构造器的参数指定DB名称（用于文件名）[12]，然后满足上述安全要求的 DB 文件会自动创建。

> [12] （未在 Android 参考中记录）由于可以在 `SQLiteOpenHelper` 实现中，将完整文件路径指定为数据库名称，因此需要注意无意中指定不能控制访问权限的地方（路径）（例如 SD 卡）。

对于如何使用，请参阅“4.5.1.1 创建/操作数据库”的具体使用方法。

使用 `Context#openOrCreateDatabase`

使用 `Context#openOrCreateDatabase` 方法创建数据库时，文件访问权应由选项指定，在这种情况下，请明确指定 `MODE_PRIVATE` 。

对于文件安排，数据库名称（用于文件名）可以像 `SQLiteOpenHelper` 一样指定，文件将在满足上述安全要求的文件路径中自动创建。 但是，也可以指定完整路径，因此有必要注意指定 SD 卡时，即使指定 `MODE_PRIVATE` ，其他应用也可以访问。

MainActivity.java（显式设定 DB 访问权的示例）

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    //Construct database
    try {
        //Create DB by setting MODE_PRIVATE
        db = Context.openOrCreateDatabase("Sample.db", MODE_PRIV
ATE, null);
    } catch (SQLException e) {
        //In case failed to construct DB, log output
        Log.e(this.getClass().toString(), getString(R.string.DAT
ABASE_OPEN_ERROR_MESSAGE));
        return;
    }
    //Omit other initial process
}
```

访问权限有三种可能的设置： `MODE_PRIVATE` ， `MODE_WORLD_READABLE` 和 `MODE_WORLD_WRITEABLE` 。 这些常量可以由或运算符一起指定。 但是，除 `API_PRIVATE` 之外的所有设置，都将在 API 级别 17 和更高版本中被弃用，并且会在 API 级别 24 和更高版本中导致安全异常。 即使对于 API 级别 15 及更早版本的应用，通常最好不要使用这些标志[13]。

> [13] `MODE_WORLD_READABLE` 和 `MODE_WORLD_WRITEABLE` 的更多信息，以及其使用的注意事项，请参见“4.6.3.2 访问目录的权限设置”。

- `MODE_PRIVATE` 只有创建者应用可以读写
- `MODE_WORLD_READABLE` 创建者应用可以读写，其他人只能读
- `MODE_WORLD_WRITEABLE` 创建者应用可以读写，其他人只能写

4.5.2.2 与其它应用共享 DB 数据时，将内容供应器用于访问控制（必需）

与其他应用共享 DB 数据的方法是，将 DB 文件创建为 `WORLD_READABLE` ， `WORLD_WRITEABLE` ，以便其他应用直接访问。 但是，此方法不能限制访问或操作数据库的应用，因此数据可以由非预期的一方（应用）读

或写。 因此，可以认为数据的机密性或一致性方面可能会出现一些问题，或者可能成为恶意软件的攻击目标。

如上所述，在 Android 中与其他应用共享数据库数据时，强烈建议使用内容供应器。 内容供应器存在一些优点，不仅从安全的角度来实现对 DB 的访问控制，而且从设计角度来看， DB 纲要结构可以隐藏到内容中。

4.5.2.3 在 DB 操作期间处理变量参数时，必需使用占位符（必需）

在防止 SQL 注入的意义上，将任意输入值并入 SQL 语句时，应使用占位符。 下面有两个方法用占位符执行 SQL。

1. 使用 `SQLiteDatabase#compileStatement()` ，获取 `SQLiteStatement` ，然后使用 `SQLiteStatement#bindString()` 或 `bindLong()` 等，将参数放置到占位符之后。
2. 在 `SQLiteDatabese` 类上调用 `execSQL()` ， `insert()` ， `update()` ， `delete()` ， `query()` ， `raw` 和 `replace()` 时，使用具有占位符的 SQL 语句。

另外，通过使用 `SQLiteDatabase#compileStatement()` 执行 `SELECT` 命令时，存在“仅获取第一个元素作为 `SELECT` 命令的结果”的限制，所以用法是有限的。

在任何一种方法中，提供给占位符的数据内容最好根据应用要求事先检查。 以下是每种方法的进一步解释。

使用 `SQLiteDatabase#compileStatement()` ：

数据以下列步骤提供给占位符：

1. 使用 `SQLiteDatabase#compileStatement()` 获取包含占位符的 SQL 语句，如 `SQLiteStatement` 。
2. 使用 `bindLong()` 和 `bindString()` 方法为创建的 `SQLiteStatement` 对象设置占位符。
3. 通过 `ExecSQLiteStatement` 对象的 `execute()` 方法执行 SQL。

DataInsertTask.java（占位符的用例）：

```
//Adding data task
public class DataInsertTask extends AsyncTask<String, Void, Void
> {

    private MainActivity mActivity;
    private SQLiteDatabase mSampleDB;

    public DataInsertTask(SQLiteDatabase db, MainActivity activi
ty) {
        mSampleDB = db;
        mActivity = activity;
    }

    @Override
    protected Void doInBackground(String... params) {
        String idno = params[0];
        String name = params[1];
        String info = params[2];
        //*** POINT 3 *** Validate the input value according the
 application requirements.
        if (!DataValidator.validateData(idno, name, info)) {
            return null;
        }
        // Adding data task
        //*** POINT 2 *** Use place holder
        String commandString = "INSERT INTO " + CommonData.TABLE
_NAME + " (idno, name, info) VALUES (?, ?, ?)";
        SQLiteStatement sqlStmt = mSampleDB.compileStatement(com
mandString);
        sqlStmt.bindString(1, idno);
        sqlStmt.bindString(2, name);
        sqlStmt.bindString(3, info);
        try {
            sqlStmt.executeInsert();
        } catch (SQLException e) {
            Log.e(DataInsertTask.class.toString(), mActivity.get
String(R.string.UPDATING_ERROR_MESSAGE));
        } finally {
            sqlStmt.close();
        }
        return null;
    }

    [...]
}
```

这是一种类型，它预先创建作为对象执行的 SQL 语句，并将参数分配给它。 执行的过程是固定的，所以没有发生 SQL 注入的可能。 另外，通过重用 `SQLiteStatement` 对象可以提高流程效率。

使用 `SQLiteDatabase` 提供的每个方法：

`SQLiteDatabase` 提供了两种类型的数据库操作方法。 一种是使用 SQL 语句，另一种是不使用 SQL 语句。 使用 SQL 语句的方法是 `SQLiteDatabase#execSQL()` / `rawQuery()` ，它以以下步骤执行。

1) 准备包含占位符的 SQL 语句。

2) 创建要分配给占位符的数据。

3) 传递 SQL 语句和数据作为参数，并为每个过程执行一个方法。

另一方面， `SQLiteDatabase#insert()/update()/delete()/query()/replace()` 是不使用 SQL 语句的方法。当使用它们时，数据应该按照以下步骤来准备。

1) 如果有数据要插入/更新到数据库，请注册到 `ContentValues` 。

2) 传递 `ContentValues` 作为参数，并为每个过程执行一个方法（例如， `SQLiteDatabase#insert()` ）

`SQLiteDatabase#insert()` （每个过程的方法的用例）：

```
private SQLiteDatabase mSampleDB;
private void addUserData(String idno, String name, String info)
{
    //Validity check of the value(Type, range), escape process
    if (!validateInsertData(idno, name, info)) {
        //If failed to pass the validation, log output
        Log.e(this.getClass().toString(), getString(R.string.VAL
IDATION_ERROR_MESSAGE));
        return;
    }
    //Prepare data to insert
    ContentValues insertValues = new ContentValues();
    insertValues.put("idno", idno);
    insertValues.put("name", name);
    insertValues.put("info", info);
    //Execute Inser
    try {
        mSampleDb.insert("SampleTable", null, insertValues);
    } catch (SQLException e) {
        Log.e(this.getClass().toString(), getString(R.string.DB_
INSERT_ERROR_MESSAGE));
        return;
    }
}
```

在这个例子中，SQL 命令不是直接写入，而是使用 `SQLiteDatabase` 提供的插入方法。 SQL 命令没有直接使用，所以在这种方法中也没有 SQL 注入的可能。

4.5.3 高级话题

4.5.3.1 在 SQL 语句的 `LIKE` 断言中使用通配符时，应该实现转义过程

当所使用的字符串包含 `LIKE` 断言的通配符（ `%` ， `_` ），作为占位符的输入值时，除非处理正确，否则它将用作通配符，因此必须根据需要事先转义处理。 通配符应该用作单个字符（ `%` 或 `_` ）时，需要转义处理。

根据下面的示例代码，使用 `ESCAPE` 子句执行实际的转义过程。

使用 `LIKE` 情况下的 `ESCAPE` 过程：

```
//Data search task
public class DataSearchTask extends AsyncTask<String, Void, Curs
or> {

    private MainActivity mActivity;
    private SQLiteDatabase mSampleDB;
    private ProgressDialog mProgressDialog;

    public DataSearchTask(SQLiteDatabase db, MainActivity activi
ty) {
        mSampleDB = db;
        mActivity = activity;
    }

    @Override
    protected Cursor doInBackground(String... params) {
        String idno = params[0];
        String name = params[1];
        String info = params[2];
        String cols[] = {"_id", "idno","name","info"};
        Cursor cur;

        [...]

        //Execute like search(partly match) with the condition o
f info
        //Point:Escape process should be performed on characters
 which is applied to wild card
        String argString = info.replaceAll("@", "@@"); // Escape
 $ in info which was received as input
        argString = argString.replaceAll("%", "@%"); // Escape %
 in info which was received as input
        argString = argString.replaceAll("_", "@_"); // Escape _
 in info which was received as input
        String selectionArgs[] = {argString};
        try {
```

```
            //Point:Use place holder
            cur = mSampleDB.query("SampleTable", cols, "info LIK
E '%' || ? || '%' ESCAPE '@'", selectionArgs, null, null, null);
        } catch (SQLException e) {
            Toast.makeText(mActivity, R.string.SERCHING_ERROR_ME
SSAGE, Toast.LENGTH_LONG).show();
            return null;
        }
        return cur;
    }

    @Override
    protected void onPostExecute(Cursor resultCur) {
        mProgressDialog.dismiss();
        mActivity.updateCursor(resultCur);
    }
}
```

4.5.3.2 不能用占位符时，在 SQL 命令中使用外部输入

当执行 SQL 语句，并且过程目标是 DB 对象，如表的创建/删除时，占位符不能用于表名的值。 基本上，数据库不应该使用外部输入的任意字符串来设计，以防占位符不能用于该值。

当由于规范或特性的限制，而无法使用占位符时，无论输入值是否危险，都应在执行前进行验证，并且需要执行必要的过程。

基本上，应该执行：

1. 使用字符串参数时，应该对于字符进行转义或引用处理。
2. 使用数字值参数时，请确认不包含数值以外的字符。
3. 用作标识符或命令时，请验证是否包含不能使用的字符以及(1)。

参考：http://www.ipa.go.jp/security/vuln/documents/website_security_sql.pdf （日文）

4.5.3.3 采取数据库非预期覆盖的对策

通过 `SQLiteOpenHelper#getReadableDatabase` 或 `getWriteableDatabase` 获取数据库实例时，通过使用任一方法 [14]，DB 将以可读/可写状态打开。 另外，与 `Context#openOrCreateDatabase` ， `SQLiteDatabase#openOrCreateDatabas` 相同。这意味着 DB 的内容可能会被应用操作，或实现中的缺陷意外覆盖。 基本上，它可以由应用规范和实现范围来支持，但是当实现仅需要读取功能的功能（如应用的搜索功能等）时，通过只读方式打开数据库，可能会简化设计或检查，从而提高应用质量，因此建议视情况而定。

> [14] `getReableDatabase()` 和 `getWritableDatabase` 可能返回同一个对象。 它的规范是，如果可写对象由于磁盘满了而无法生成，它将返回只读对象。 （ `getWritableDatabase()` 会在磁盘满了的情况下产生错误）

特别是，通过对 `SQLiteDatabase#openDatabase` 指定 `OPEN_READONLY` 打开数据库。

以只读打开数据库：

```
[...]
// Open DB(DB should be created in advance)
SQLiteDatabase db
    = SQLiteDatabase.openDatabase(SQLiteDatabase.getDatabasePath(
"Sample.db"), null, OPEN_READONLY);
```

参考：http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#getReadableDatabase()

4.5.3.4 根据应用需求，验证 DB 的输入输出数据的有效性

SQLite 是类型容错的数据库，它可以将字符类型数据存储到在 DB 中声明为整数的列中。 对于数据库中的数据，包括数值类型的所有数据都作为纯文本的字符数据存储在数据库中。 所以搜索字符串类型，可以对整数类型的列执行（ `LIKE '%123%'` 等）。此外，由于在某些情况下，可以输入超过限制的数据，所以对 SQLite 中的值（有效性验证）的限制是不可信的，例如 `VARCHAR(100)` 。

因此，使用 SQLite 的应用需要非常小心 DB 的这种特性，并且有必要根据应用需求采取措施，不要将意外的数据存储到数据库，或不要获取意外的数据。 对策是以下两点。

1. 在数据库中存储数据时，请确认类型和长度是否匹配。
2. 从数据库中获取值时，验证数据是否超出假定的类型和长度。

下面是个代码示例，它验证了输入值是否大于 1。

```
public class MainActivity extends Activity {

    [...]

    //Process for adding
    private void addUserData(String idno, String name, String in
fo) {
        //Check for No
        if (!validateNo(idno, CommonData.REQUEST_NEW)) {
            return;
        }
        //Inserting data process
        DataInsertTask task = new DataInsertTask(mSampleDbyhis);
        task.execute(idno, name, info);
    }

    [...]

    private boolean validateNo(String idno, int request) {
        if (idno == null || idno.length() == 0) {
            if (request == CommonData.REQUEST_SEARCH) {
                //When search process, unspecified is considered
 as OK.
                return true;
            } else {
                //Other than search process, null and blank are
error.
                Toast.makeText(this, R.string.IDNO_EMPTY_MESSAGE
, Toast.LENGTH_LONG).show();
                return false;
            }
        }
        //Verify that it's numeric character
        try {
            // Value which is more than 1
            if (!idno.matches("[1-9][0-9]*")) {
                //In case of not numeric character, error
                Toast.makeText(this, R.string.IDNO_NOT_NUMERIC_M
ESSAGE, Toast.LENGTH_LONG).show();
                return false;
            }
        } catch (NullPointerException e) {
            //It never happen in this case
            return false;
        }
        return true;
    }

    [...]
}
```

4.5.3.5 考虑 -- 储存在数据库中的数据

在 SQLite 视线中，将数据储存到文件是这样：

- 所有包含数值类型的数据，都将作为纯文本的字符数据存储在 DB 文件中。
- 执行 DB 的数据删除时，数据本身不会从 DB 文件中删除。（只添加删除标记。）
- 更新数据时，更新前的数据未被删除，仍保留在数据库文件中。

因此，“必须”删除的信息仍可能保留在 DB 文件中。即使在这种情况下，也要根据本指导手册采取对策，并且启用 Android 安全功能时，数据/文件可能不会被第三方直接访问，包括其他应用。但考虑到通过绕过 Android 的保护系统（如 root 权限）选取文件的情况，如果存储了对业务有巨大影响的数据，则应考虑不依赖于 Android 保护系统的数据保护。

由于上述原因，需要保护的重要数据，不应该存储在 SQLite 数据库中，即使设备取得了 root 权限。在需要存储重要数据的情况下，有必要采取对策或加密整个数据库。

当需要加密时，有许多问题超出了本指南的范围，比如处理用于加密或代码混淆的密钥，所以目前建议，在开发处理数据的应用，数据对业务有巨大影响时咨询专家。请参考“4.5.3.6 [参考] 加密 SQLite 数据库（Android `SQLCipher`）”，这里介绍加密数据库的库。

4.5.3.6 [参考] 加密 SQLite 数据库（Android `SQLCipher`）

`SQLCipher` 是为数据库提供透明 256 位 AES 加密的 SQLite 扩展。它是开源的（BSD 许可证），由 Zetetic LLC 维护/管理。在移动世界中，`SQLCipher` 广泛用于诺基亚/ QT，苹果的 iOS。

Android 项目的 `SQLCipher` 旨在支持 Android 环境中的 SQLite 数据库的标准集成加密。通过为 `SQLCipher` 创建标准 SQLite 的 API，开发人员可以使用加密的数据库和平常一样的编码。

参考：https://guardianproject.info/code/sqlcipher/。

如何使用：

应用开发者可以通过以下三个步骤使用 `SQLCipher`。

1. 在应用的 `lib` 目录中找到 `sqlcipher.jar`，`libdatabase_sqlcipher.so`，`libsqlcipher_andr` 和 `libstlport_shared.so`。
2. 对于所有源文件，将所有 `android.database.sqlite.*` 更改为 `info.guardianproject.database.sqlite.*`，它们由 `import` 指定。另外，`android.database.Cursor` 可以照原样使用。
3. 在 `onCreate()` 中初始化数据库，打开数据库时设置密码。

简单的代码示例：

```
SQLiteDatabase.loadLibs(this); // First, Initialize library by u
sing context.
SQLiteOpenHelper.getWRITEABLEDatabase(passwoed): // Parameter is
 password(Suppose that it's string type and It's got in a secure
 way.)
```

在撰写本文时，Android 版 `SQLCipher` 是 1.1.0 版，现在正在开发 2.0.0 版，现在已经公布了 RC4。 就过去在 Android 中的使用和 API 的稳定性而言，有必要稍后进行验证，但目前还可以看做 SQLite 的加密解决方案，它可以在 Android 中使用。

库的结构

下列 SDK 中包含的文件是使用 `SQLCipher` 所必须的。

- `assets/icudt46l.zip` 2,252KB

当 `icudt46l.dat` 不存在于 `/system/usr/icu/` 下及其早期版本时，这是必需的。 当找不到 `icudt46l.dat` 时，此 zip 需要解压缩并使用。

- `libs/armeabi/libdatabase_sqlcipher.so` 44KB
- `libs/armeabi/libsqlcipher_android.so` 1,117KB
- `libs/armeabi/libstlport_shared.so` 555KB

本地库，它在 `SQLCipher` 首次加载（调用 `SQLiteDatabase#loadLibs()` ）时被读取。

- `libs/commons-codec.jar` 46KB
- `libs/guava-r09.jar` 1,116KB
- `libs/sqlcipher.jar` 102KB

Java 库调用本地库。 `sqlcipher.jar` 是主要的，其它的由 `sqlcipher.jar` 引用。

总共大约 5.12MB。但是，当 `icudt46l.zip` 解压时，总共大约 7MB。

4.6 处理文件

根据 Android 安全设计理念，文件仅用于信息持久化和临时保存（缓存），原则上它应该是私有的。 在应用之间交换信息不应该直接通过文件，而应该通过应用间的连接系统（如内容供应器或服务）来交换。 通过使用此功能，可以实现应用间访问控制。

由于无法在 SD 卡等外部存储设备上执行足够的访问控制，因此文件应限制仅在必要时通过功能方式使用，例如处理大型文件，或将信息传输到其他位置时（PC 等等）。 基本上，包含敏感信息的文件不应保存在外部存储设备中。 在需要将敏感信息保存在外部设备文件中的情况下，需要采取加密等对策，但这里没有提及。

4.6.1 示例代码

如上所述，文件原则上应该是私有的。 但是，由于某些原因，有时文件应该由其他应用直接读写。 按照安全角度分类和比较中文件类型如表 4.6-1 所示。 它们根据文件存储位置或其他应用的访问权限分为四类。 下面展示了每个文件类别的示例代码，并在其中添加了每个的解释。

表 4.6-1 按照安全角度的文件类别和比较

文件类别	其它应用的访问权限	储存位置	概述
私有文件	NA	应用目录中	（1）只能在应用中读写，（2）可以处理敏感数据，（3）文件原则上应该是这个类型
只读公共文件	读	应用目录中	（1）其它应用和用户可读，（2）可以处理公开给应用外部的信息
读写公共文件	读写	应用目录中	（1）其它应用和用户可以读写，（2）从安全和应用设计角度来看，不应该使用
外部存储设备（读写文件）	读写	外部存储设备，例如 SD 卡	（1）没有访问控制，（2）其它应用和用户总是可以读写或删除文件，（3）应该以最小需求使用，（4）可以处理很大的文件

4.6.1.1 使用私有文件

这种情况下使用的文件，只能在同一个应用中读取/写入，并且这是使用文件的一种非常安全的方式。 原则上，无论存储在文件中的信息是否是公开的，尽可能使用私有文件，当与其他应用交换必要的信息时，应该使用另一个 Android 系统（内容供应器，服务）来完成。

要点：

1) 文件必须在应用目录中创建。

2) 文件的访问权限必须设置为私有模式，以免其他应用使用。

3) 可以存储敏感信息。

4) 对于存储在文件中的信息，请仔细和安全地处理文件数据。

PrivateFileActivity.java

```
package org.jssec.android.file.privatefile;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
public class PrivateFileActivity extends Activity {
    private TextView mFileView;
    private static final String FILE_NAME = "private_file.dat";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.file);
        mFileView = (TextView) findViewById(R.id.file_view);
    }

    /**
    * Create file process
    *
    * @param view
    */
    public void onCreateFileClick(View view) {
        FileOutputStream fos = null;
        try {
            // *** POINT 1 *** Files must be created in applicat
ion directory.
            // *** POINT 2 *** The access privilege of file must
```

```
 be set private mode in order not to be used by other applicatio
ns.
            fos = openFileOutput(FILE_NAME, MODE_PRIVATE);
            // *** POINT 3 *** Sensitive information can be stor
ed.
            // *** POINT 4 *** Regarding the information to be s
tored in files, handle file data carefully and securely.
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
            fos.write(new String("Not sensotive information (Fil
e Activity)¥n").getBytes());
        } catch (FileNotFoundException e) {
            mFileView.setText(R.string.file_view);
        } catch (IOException e) {
            android.util.Log.e("PrivateFileActivity", "failed to
 read file");
        } finally {
            if (fos != null) {
                try {
                    fos.close();
                } catch (IOException e) {
                    android.util.Log.e("PrivateFileActivity", "f
ailed to close file");
                }
            }
        }
        finish();
    }

    /**
     * Read file process
     *
     * @param view
     */
    public void onReadFileClick(View view) {
        FileInputStream fis = null;
        try {
            fis = openFileInput(FILE_NAME);
            byte[] data = new byte[(int) fis.getChannel().size()
];

            fis.read(data);
            String str = new String(data);
            mFileView.setText(str);
        } catch (FileNotFoundException e) {
            mFileView.setText(R.string.file_view);
        } catch (IOException e) {
            android.util.Log.e("PrivateFileActivity", "failed to
 read file");
        } finally {
            if (fis != null) {
                try {
                    fis.close();
                } catch (IOException e) {
```

```
                    android.util.Log.e("PrivateFileActivity", "f
ailed to close file");
                }
            }
        }
    }

    /**
     * Delete file process
     *
     * @param view
     */
    public void onDeleteFileClick(View view) {
        File file = new File(this.getFilesDir() + "/" + FILE_NAM
E);
        file.delete();
        mFileView.setText(R.string.file_view);
    }
}
```

PrivateUserActivity.java

```
package org.jssec.android.file.privatefile;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class PrivateUserActivity extends Activity {

    private TextView mFileView;
    private static final String FILE_NAME = "private_file.dat";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.user);
        mFileView = (TextView) findViewById(R.id.file_view);
    }

    private void callFileActivity() {
        Intent intent = new Intent();
        intent.setClass(this, PrivateFileActivity.class);
        startActivity(intent);
    }
```

```
    /**
     * Call file Activity process
     *
     * @param view
     */
    public void onCallFileActivityClick(View view) {
        callFileActivity();
    }

    /**
     * Read file process
     *
     * @param view
     */
    public void onReadFileClick(View view) {
        FileInputStream fis = null;
        try {
            fis = openFileInput(FILE_NAME);
            byte[] data = new byte[(int) fis.getChannel().size()
];
            fis.read(data);
            // *** POINT 4 *** Regarding the information to be s
tored in files, handle file data carefully and securely.
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
            String str = new String(data);
            mFileView.setText(str);
        } catch (FileNotFoundException e) {
            mFileView.setText(R.string.file_view);
        } catch (IOException e) {
            android.util.Log.e("PrivateUserActivity", "failed to
 read file");
        } finally {
            if (fis != null) {
                try {
                    fis.close();
                } catch (IOException e) {
                    android.util.Log.e("PrivateUserActivity", "f
ailed to close file");
                }
            }
        }
    }

    /**
     * Rewrite file process
     *
     * @param view
     */
    public void onWriteFileClick(View view) {
        FileOutputStream fos = null;
        try {
```

```
            // *** POINT 1 *** Files must be created in applicat
ion directory.
            // *** POINT 2 *** The access privilege of file must
 be set private mode in order not to be used by other applicatio
ns.
            fos = openFileOutput(FILE_NAME, MODE_APPEND);
            // *** POINT 3 *** Sensitive information can be stor
ed.
            // *** POINT 4 *** Regarding the information to be s
tored in files, handle file data carefully and securely.
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
            fos.write(new String("Sensitive information (User Ac
tivity)¥n").getBytes());
        } catch (FileNotFoundException e) {
            mFileView.setText(R.string.file_view);
        } catch (IOException e) {
            android.util.Log.e("PrivateUserActivity", "failed to
 read file");
        } finally {
            if (fos != null) {
                try {
                    fos.close();
                } catch (IOException e) {
                    android.util.Log.e("PrivateUserActivity", "f
ailed to close file");
                }
            }
        }
        callFileActivity();
    }
}
```

4.6.1.2 使用公共只读文件

这是使用文件向未指定的大量应用公开内容的情况。 如果通过遵循以下几点来实现，那么它也是比较安全的文件使用方法。 请注意，在 API 级别 1 7及更高版本中，不推荐使用 `MODE_WORLD_READABLE` 变量来创建公共文件，并且在 API 级别 24 及更高版本中，会触发安全异常; 因此使用内容供应器的文件共享方法更可取。

要点：

1) 文件必须在应用目录中创建。

2) 文件的访问权限必须设置为其他应用只读。

3) 敏感信息不得存储。

4) 对于要存储在文件中的信息，请仔细和安全地处理文件数据。

PublicFileActivity.java

```
package org.jssec.android.file.publicfile.readonly;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class PublicFileActivity extends Activity {

    private TextView mFileView;
    private static final String FILE_NAME = "public_file.dat";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.file);
        mFileView = (TextView) findViewById(R.id.file_view);
    }

    /**
    * Create file process
    *
    * @param view
    */
    public void onCreateFileClick(View view) {
        FileOutputStream fos = null;
        try {
```

```
            // *** POINT 1 *** Files must be created in applicat
ion directory.
            // *** POINT 2 *** The access privilege of file must
 be set to read only to other applications.
            // (MODE_WORLD_READABLE is deprecated API Level 17,
            // don't use this mode as much as possible and excha
nge data by using ContentProvider().)
            fos = openFileOutput(FILE_NAME, MODE_WORLD_READABLE)
;
            // *** POINT 3 *** Sensitive information must not be
 stored.
            // *** POINT 4 *** Regarding the information to be s
tored in files, handle file data carefully and securely.
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
            fos.write(new String("Not sensitive information (Pub
lic File Activity)¥n").getBytes());
        } catch (FileNotFoundException e) {
            mFileView.setText(R.string.file_view);
        } catch (IOException e) {
            android.util.Log.e("PublicFileActivity", "failed to
read file");
        } finally {
            if (fos != null) {
                try {
                    fos.close();
                } catch (IOException e) {
                    android.util.Log.e("PublicFileActivity", "fa
iled to close file");
                }
            }
        }
        finish();
    }

    /**
    * Read file process
    *
    * @param view
    */
    public void onReadFileClick(View view) {
        FileInputStream fis = null;
        try {
            fis = openFileInput(FILE_NAME);
            byte[] data = new byte[(int) fis.getChannel().size()
];

            fis.read(data);
            String str = new String(data);
            mFileView.setText(str);
        } catch (FileNotFoundException e) {
            mFileView.setText(R.string.file_view);
        } catch (IOException e) {
            android.util.Log.e("PublicFileActivity", "failed to
```

```
read file");
        } finally {
            if (fis != null) {
                try {
                    fis.close();
                } catch (IOException e) {
                    android.util.Log.e("PublicFileActivity", "fa
iled to close file");
                }
            }
        }
    }

    /**
     * Delete file process
     *
     * @param view
     */
    public void onDeleteFileClick(View view) {
        File file = new File(this.getFilesDir() + "/" + FILE_NAM
E);
        file.delete();
        mFileView.setText(R.string.file_view);
    }
}
```

PublicUserActivity.java

```
package org.jssec.android.file.publicuser.readonly;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import android.app.Activity;
import android.content.ActivityNotFoundException;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager.NameNotFoundException;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class PublicUserActivity extends Activity {

    private TextView mFileView;
    private static final String TARGET_PACKAGE = "org.jssec.andr
oid.file.publicfile.readonly";
    private static final String TARGET_CLASS = "org.jssec.androi
d.file.publicfile.readonly.PublicFileActivity";
```

```
    private static final String FILE_NAME = "public_file.dat";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.user);
        mFileView = (TextView) findViewById(R.id.file_view);
    }

    private void callFileActivity() {
        Intent intent = new Intent();
        intent.setClassName(TARGET_PACKAGE, TARGET_CLASS);
        try {
            startActivity(intent);
        } catch (ActivityNotFoundException e) {
            mFileView.setText("(File Activity does not exist)");
        }
    }

    /**
    * Call file Activity process
    *
    * @param view
    */
    public void onCallFileActivityClick(View view) {
        callFileActivity();
    }

    /**
    * Read file process
    *
    * @param view
    */
    public void onReadFileClick(View view) {
        FileInputStream fis = null;
        try {
            File file = new File(getFilesPath(FILE_NAME));
            fis = new FileInputStream(file);
            byte[] data = new byte[(int) fis.getChannel().size()
];
            fis.read(data);
            // *** POINT 4 *** Regarding the information to be s
tored in files, handle file data carefully and securely.
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
            String str = new String(data);
            mFileView.setText(str);
        } catch (FileNotFoundException e) {
            android.util.Log.e("PublicUserActivity", "no file");
        } catch (IOException e) {
            android.util.Log.e("PublicUserActivity", "failed to
read file");
        } finally {
```

```
            if (fis != null) {
                try {
                    fis.close();
                } catch (IOException e) {
                    android.util.Log.e("PublicUserActivity", "fa
iled to close file");
                }
            }
        }
    }

    /**
    * Rewrite file process
    *
    * @param view
    */
    public void onWriteFileClick(View view) {
        FileOutputStream fos = null;
        boolean exception = false;
        try {
            File file = new File(getFilesPath(FILE_NAME));
            // Fail to write in. FileNotFoundException occurs.
            fos = new FileOutputStream(file, true);
            fos.write(new String("Not sensitive information (Pub
lic User Activity)¥n").getBytes());
        } catch (IOException e) {
            mFileView.setText(e.getMessage());
            exception = true;
        } finally {
            if (fos != null) {
                try {
                    fos.close();
                } catch (IOException e) {
                    exception = true;
                }
            }
        }
        if (!exception)
            callFileActivity();
    }

    private String getFilesPath(String filename) {
        String path = "";
        try {
            Context ctx = createPackageContext(TARGET_PACKAGE,
            Context.CONTEXT_RESTRICTED);
            File file = new File(ctx.getFilesDir(), filename);
            path = file.getPath();
        } catch (NameNotFoundException e) {
            android.util.Log.e("PublicUserActivity", "no file");
        }
        return path;
    }
```

```
}
```

4.6.1.3 创建公共读写文件

这是一种文件用法，它允许未指定的大量应用的读写访问。

未指定的大量应用可以读写，意思不用多说了。 恶意软件也可以读取和写入，因此数据的可信度和安全性将永远不会得到保证。 另外，即使在没有恶意的情况下，也不能控制文件中的数据格式或写入的时间。 所以这种类型的文件在功能方面几乎不实用。

如上所述，从安全性和应用设计的角度来看，不可能安全地使用读写文件，因此应该避免使用读写文件。

要点：

不要创建允许来自其他应用的读写操作的文件。

4.6.1.4 使用外部存储器（公共读写）文件

将文件存储在 SD 卡等外部存储器中时，就是这种情况。当存储比较庞大的信息（放置从 Web 下载的文件）或者将信息带出到外部时（备份等）时，应该使用它。

对于未指定的大量应用，“外部存储器文件（公共读写）”与“公共读写文件“有相同特性。另外，对于声明使用 `android.permission.WRITE_EXTERNAL_STORAGE` 权限的应用，它和“公共读写文件”具有相同的特性。因此，应尽可能减少“外部存储器（公共读写）文件”的使用。

按照 Android 应用的惯例，备份文件很可能是在外部存储器中创建的。但是，如上所述，外部存储器中的文件存在被其他应用（包括恶意软件）篡改/删除的风险。因此，在输出备份的应用中，为了最小化应用规范或设计方面的风险，一些设计是必要的，例如显示“尽快将备份文件复制到 PC 等安全位置”。

要点：

1) 不得存储敏感信息。

2) 文件必须存储在每个应用的唯一目录中。

3) 对于要存储在文件中的信息，请仔细和安全地处理文件数据。

4) 请求应用的文件写入应该按照规范禁止。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.file.externalfile" >
    <!-- declare android.permission.WRITE_EXTERNAL_STORAGE permi
ssion to write to the external strage --
    >
    <!-- In Android 4.4 (API Level 19) and later, the applicatio
n, which read/write only files in its sp
    ecific
    directories on external storage media, need not to require t
he permission and it should declare
    the maxSdkVersion -->
    <uses-permission android:name="android.permission.WRITE_EXTE
RNAL_STORAGE"
    android:maxSdkVersion="18"/>
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:allowBackup="false" >
        <activity
            android:name=".ExternalFileActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

ExternalFileActivity.java

```
package org.jssec.android.file.externalfile;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class ExternalFileActivity extends Activity {
```

```
    private TextView mFileView;
    private static final String TARGET_TYPE = "external";
    private static final String FILE_NAME = "external_file.dat";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.file);
        mFileView = (TextView) findViewById(R.id.file_view);

    }
    /**
    * Create file process
    *
    * @param view
    */
    public void onCreateFileClick(View view) {
        FileOutputStream fos = null;
        try {
            // *** POINT 1 *** Sensitive information must not be
 stored.
            // *** POINT 2 *** Files must be stored in the uniqu
e directory per application.
            File file = new File(getExternalFilesDir(TARGET_TYPE
), FILE_NAME);
            fos = new FileOutputStream(file, false);
            // *** POINT 3 *** Regarding the information to be s
tored in files, handle file data carefully and securely.
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
            fos.write(new String("Non-Sensitive Information(Exte
rnalFileActivity)¥n")
            .getBytes());
        } catch (FileNotFoundException e) {
            mFileView.setText(R.string.file_view);
        } catch (IOException e) {
            android.util.Log.e("ExternalFileActivity", "failed t
o read file");
        } finally {
            if (fos != null) {
                try {
                    fos.close();
                } catch (IOException e) {
                    android.util.Log.e("ExternalFileActivity", "
failed to close file");
                }
            }
        }
        finish();
    }

    /**
    * Read file process
```

```
     *
     * @param view
     */
    public void onReadFileClick(View view) {
        FileInputStream fis = null;
        try {
            File file = new File(getExternalFilesDir(TARGET_TYPE
), FILE_NAME);
            fis = new FileInputStream(file);
            byte[] data = new byte[(int) fis.getChannel().size()
];
            fis.read(data);
            // *** POINT 3 *** Regarding the information to be s
tored in files, handle file data carefully and securely.
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
            String str = new String(data);
            mFileView.setText(str);
        } catch (FileNotFoundException e) {
            mFileView.setText(R.string.file_view);
        } catch (IOException e) {
            android.util.Log.e("ExternalFileActivity", "failed t
o read file");
        } finally {
            if (fis != null) {
                try {
                    fis.close();
                } catch (IOException e) {
                    android.util.Log.e("ExternalFileActivity", "
failed to close file");
                }
            }
        }
    }

    /**
     * Delete file process
     *
     * @param view
     */
    public void onDeleteFileClick(View view) {
        File file = new File(getExternalFilesDir(TARGET_TYPE), F
ILE_NAME);
        file.delete();
        mFileView.setText(R.string.file_view);
    }
}
```

使用的示例代码：

ExternalFileUser.java

```
package org.jssec.android.file.externaluser;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.ActivityNotFoundException;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager.NameNotFoundException;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class ExternalUserActivity extends Activity {

    private TextView mFileView;
    private static final String TARGET_PACKAGE = "org.jssec.andr
oid.file.externalfile";
    private static final String TARGET_CLASS = "org.jssec.androi
d.file.externalfile.ExternalFileActivity";
    private static final String TARGET_TYPE = "external";
    private static final String FILE_NAME = "external_file.dat";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.user);
        mFileView = (TextView) findViewById(R.id.file_view);
    }
    private void callFileActivity() {
        Intent intent = new Intent();
        intent.setClassName(TARGET_PACKAGE, TARGET_CLASS);
        try {
            startActivity(intent);
        } catch (ActivityNotFoundException e) {
            mFileView.setText("(File Activity does not exist)");
        }
    }

    /**
    * Call file Activity process
    *
    * @param view
    */
    public void onCallFileActivityClick(View view) {
        callFileActivity();
    }

    /**
```

```
     * Read file process
     *
     * @param view
     */
    public void onReadFileClick(View view) {
        FileInputStream fis = null;
        try {
            File file = new File(getFilesPath(FILE_NAME));
            fis = new FileInputStream(file);
            byte[] data = new byte[(int) fis.getChannel().size()
];
            fis.read(data);
            // *** POINT 3 *** Regarding the information to be s
tored in files, handle file data carefully and securely.
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
            String str = new String(data);
            mFileView.setText(str);
        } catch (FileNotFoundException e) {
            mFileView.setText(R.string.file_view);
        } catch (IOException e) {
            android.util.Log.e("ExternalUserActivity", "failed t
o read file");
        } finally {
            if (fis != null) {
                try {
                    fis.close();
                } catch (IOException e) {
                    android.util.Log.e("ExternalUserActivity", "
failed to close file");
                }
            }
        }
    }

    /**
     * Rewrite file process
     *
     * @param view
     */
    public void onWriteFileClick(View view) {
        // *** POINT 4 *** Writing file by the requesting applic
ation should be prohibited as the specification.
        // Application should be designed supposing malicious ap
plication may overwrite or delete file.
        final AlertDialog.Builder alertDialogBuilder = new Alert
Dialog.Builder(this);
        alertDialogBuilder.setTitle("POINT 4");
        alertDialogBuilder.setMessage("Do not write in calling a
pplication.");
        alertDialogBuilder.setPositiveButton("OK",
        new DialogInterface.OnClickListener() {
            @Override
```

```
                public void onClick(DialogInterface dialog, int whic
h) {
                    callFileActivity();
                }
            });
            alertDialogBuilder.create().show();
        }

        private String getFilesPath(String filename) {
            String path = "";
            try {
                Context ctx = createPackageContext(TARGET_PACKAGE,
                Context.CONTEXT_IGNORE_SECURITY);
                File file = new File(ctx.getExternalFilesDir(TARGET_
TYPE), filename);
                path = file.getPath();
            } catch (NameNotFoundException e) {
                android.util.Log.e("ExternalUserActivity", "no file"
);
            }
            return path;
        }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.file.externaluser" >
    <!-- In Android 4.0.3 (API Level 14) and later, the permissi
on for reading external storages
    has been defined and the application should decalre that it
requires the permission.
    In fact in Android 4.4 (API Level 19) and later, that must b
e declared to read other directories
    than the package specific directories. -->
    <uses-permission android:name="android.permission.READ_EXTER
NAL_STORAGE" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:allowBackup="false" >
        <activity
            android:name=".ExternalUserActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

4.6.2 规则书

遵循以下规则：

4.6.2.1 文件原则上必须创建为私有（必需）

如“4.6 处理文件”和“4.6.1.3 使用公共读/写文件”所述，无论要存储的信息的内容如何，原则上都应该将文件设置为私有。 从 Android 安全角度来看，交换信息及其访问控制应该在 Android 系统中完成，如内容供应器和服务，并且如果存在不可能的因素，则应该考虑由文件访问权限作为替代方法。

请参阅每个文件类型的示例代码和以下规则条目。

4.6.2.2 禁止创建允许来自其他应用的读写访问的文件（必需）

如“4.6.1.3 使用公共读/写文件”中所述，当允许其他应用读取/写入文件时，存储在文件中的信息无法控制。 因此，从安全和功能/设计的角度来看，不应该用公共读/写文件共享信息。

4.6.2.3 使用存储在外部存储器如 SD 卡）的文件，应该尽可能最小（必需）

如“4.6.1.4 使用外部存储器（公共读写）文件”中所述，出于安全和功能的考虑，将文件存储在外部存储器（如 SD 卡）中，会导致潜在的问题。 另一方面，与应用目录相比，SD 卡可以处理更大范围的文件，并且这是可以用于将数据带出到应用之外的唯一存储器。 所以，可能有很多情况下必须使用它，取决于应用的规范。

将文件存储在外部存储器中时，考虑到未指定的大量应用和用户可以读/写/删除文件，所以有必要考虑以下各点以及示例代码中提及的要点，来设计应用。

- 原则上，敏感信息不应保存在外部存储器的文件中。
- 将敏感信息保存在外部存储器的文件中时，应将其加密。
- 将文件保存在外部存储器时，如果被其他应用或用户篡改，将会出现问题，应该用电子签名保存。
- 当读入外部存储器中的文件时，请在验证读取的数据安全性后使用数据。
- 应该这样设计应用，假设外部存储器中的文件始终可以被删除。

请参考“4.6.2.4 应用应该在考虑文件范围的情况下设计”。

4.6.2.4 应用应该在考虑文件范围的情况下设计（必需）

保存在应用目录中的数据，被以下用户操作删除。 它与应用的范围是一致的，并且与应用的范围相比，它的独特之处在于它比应用的范围小。

- 卸载应用
- 删除每个应用的数据和缓存（设置=>应用=>选择目标应用）

保存在外部存储器中的文件，如 SD 卡，文件的范围比应用的范围长。 另外，还需要考虑以下情况。

- 文件由用户删除
- 取出/替换/取消挂载 SD 卡
- 文件由恶意软件删除

如上所述，由于文件范围取决于文件的保存位置而有所不同，不仅从保护敏感信息的角度，而且从实现应用的正确行为的角度，有必要选择文件保存位置。

4.6.3 高级话题

4.6.3.1 通过文件描述符的文件共享

有一种方法可以通过文件描述符共享文件，而不是让其他应用访问公共文件。 此方法可用在内容供应器和服务中。 对方的应用可以通过文件描述符读取/写入文件，这些文件描述符通过在内容供应器或服务中，打开私人文件来获得。

其他应用直接访问文件的共享方式，与文件描述符的共享方式的比较如下表 4.6-2。 优点是访问权限的变化，以及允许访问的应用范围。 特别是从安全角度来看，这是一个很大的优点，可以详细控制允许访问的应用。

表 4.6-2 应用内文件共享方式的比较

文件共享方式	验证或者访问权限设置	允许访问的应用范围
允许其他应用直接访问的文件共享	读、写、读写	给予所有应用同等访问权限
通过文件描述符的文件共享	读、写、仅添加、读写、读+添加	可以控制是否将权限授予应用，它们尝试独立和暂时访问内容供应器和服务。

在上述两种文件共享方法中，这是很常见的，因为向其他应用提供文件写入权限时，文件内容的完整性很难得到保证。 当多个应用并行写入时，可能会破坏文件内容的数据结构，导致应用无法正常工作。 因此，在与其他应用共享文件时，只允许只读权限。

以下是通过内容供应器的文件共享的实现示例，及其示例代码。

要点：

1) 源应用是内部应用，因此可以保存敏感信息。

2) 即使是由内部的内容供应器产生的结果，也要验证结果数据的安全性。

InhouseProvider.java

```
package org.jssec.android.file.inhouseprovider;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import org.jssec.android.shared.SigPerm;
import org.jssec.android.shared.Utils;
import android.content.ContentProvider;
```

```java
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
import android.os.ParcelFileDescriptor;

public class InhouseProvider extends ContentProvider {

    private static final String FILENAME = "sensitive.txt";
    // In-house signature permission
    private static final String MY_PERMISSION = "org.jssec.andro
id.file.inhouseprovider.MY_PERMISSION";
    // In-house certificate hash value
    private static String sMyCertHash = null;

    private static String myCertHash(Context context) {
        if (sMyCertHash == null) {
            if (Utils.isDebuggable(context)) {
                // Certificate hash value of debug.keystore "and
roiddebugkey"
                sMyCertHash = "0EFB7236 328348A9 89718BAD DF57F5
44 D5CCB4AE B9DB34BC 1E29DD26 F77C8255";
            } else {
                // Certificate hash value of keystore "my compan
y key"
                sMyCertHash = "D397D343 A5CBC10F 4EDDEB7C A10062
DE 5690984F 1FB9E88B D7B3A7C2 42E142CA";
            }
        }
        return sMyCertHash;
    }

    @Override
    public boolean onCreate() {
        File dir = getContext().getFilesDir();
        FileOutputStream fos = null;
        try {
            fos = new FileOutputStream(new File(dir, FILENAME));
            // *** POINT 1 *** The source application is In hous
e application, so sensitive information can be saved.
            fos.write(new String("Sensitive information").getByt
es());
        } catch (IOException e) {
            android.util.Log.e("InhouseProvider", "failed to rea
d file");
        } finally {
            try {
                fos.close();
            } catch (IOException e) {
                android.util.Log.e("InhouseProvider", "failed to
 close file");
            }
        }
```

```
        return true;
    }

    @Override
    public ParcelFileDescriptor openFile(Uri uri, String mode)
        throws FileNotFoundException {
        // Verify that in-house-defined signature permission is
defined by in-house application.
        if (!SigPerm.test(getContext(), MY_PERMISSION, myCertHas
h(getContext()))) {
            throw new SecurityException(
                "In-house-defined signature permission is not de
fined by in-house application.");
        }
        File dir = getContext().getFilesDir();
        File file = new File(dir, FILENAME);
        // Always return read-only, since this is sample
        int modeBits = ParcelFileDescriptor.MODE_READ_ONLY;
        return ParcelFileDescriptor.open(file, modeBits);
    }

    @Override
    public String getType(Uri uri) {
        return "";
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String sel
ection,
        String[] selectionArgs, String sortOrder) {
        return null;
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        return null;
    }

    @Override
    public int update(Uri uri, ContentValues values, String sele
ction,
        String[] selectionArgs) {
        return 0;
    }

    @Override
    public int delete(Uri uri, String selection, String[] select
ionArgs) {
        return 0;
    }
}
```

InhouseUserActivity.java

```
package org.jssec.android.file.inhouseprovideruser;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import org.jssec.android.shared.PkgCert;
import org.jssec.android.shared.SigPerm;
import org.jssec.android.shared.Utils;
import android.app.Activity;
import android.content.Context;
import android.content.pm.PackageManager;
import android.content.pm.ProviderInfo;
import android.net.Uri;
import android.os.Bundle;
import android.os.ParcelFileDescriptor;
import android.view.View;
import android.widget.TextView;

public class InhouseUserActivity extends Activity {

    // Content Provider information of destination (requested pr
ovider)
    private static final String AUTHORITY = "org.jssec.android.f
ile.inhouseprovider";
    // In-house signature permission
    private static final String MY_PERMISSION = "org.jssec.andro
id.file.inhouseprovider.MY_PERMISSION";
    // In-house certificate hash value
    private static String sMyCertHash = null;

    private static String myCertHash(Context context) {
        if (sMyCertHash == null) {
            if (Utils.isDebuggable(context)) {
                // Certificate hash value of debug.keystore "and
roiddebugkey"
                sMyCertHash = "0EFB7236 328348A9 89718BAD DF57F5
44 D5CCB4AE B9DB34BC 1E29DD26 F77C8255";
            } else {
                // Certificate hash value of keystore "my compan
y key"
                sMyCertHash = "D397D343 A5CBC10F 4EDDEB7C A10062
DE 5690984F 1FB9E88B D7B3A7C2 42E142CA";
            }
        }
        return sMyCertHash;
    }

    // Get package name of destination (requested) content provi
der.
    private static String providerPkgname(Context context, Strin
```

```
g authority) {
        String pkgname = null;
        PackageManager pm = context.getPackageManager();
        ProviderInfo pi = pm.resolveContentProvider(authority, 0
);
        if (pi != null)
        pkgname = pi.packageName;
        return pkgname;
    }

    public void onReadFileClick(View view) {
        logLine("[ReadFile]");
        // Verify that in-house-defined signature permission is
defined by in-house application.
        if (!SigPerm.test(this, MY_PERMISSION, myCertHash(this))
) {
            logLine(" In-house-defined signature permission is n
ot defined by in-house application.");
            return;
        }
        // Verify that the certificate of destination (requested
) content provider application is in-house certificate.
        String pkgname = providerPkgname(this, AUTHORITY);
        if (!PkgCert.test(this, pkgname, myCertHash(this))) {
            logLine(" Destination (Requested) Content Provider i
s not in-house application.");
            return;
        }
        // Only the information which can be disclosed to in-hou
se only content provider application, can be included in a reque
st.
        ParcelFileDescriptor pfd = null;
        try {
            pfd = getContentResolver().openFileDescriptor(
            Uri.parse("content://" + AUTHORITY), "r");
        } catch (FileNotFoundException e) {
            android.util.Log.e("InhouseUserActivity", "no file")
;
        }
        if (pfd != null) {
            FileInputStream fis = new FileInputStream(pfd.getFil
eDescriptor());
            if (fis != null) {
                try {
                    byte[] buf = new byte[(int) fis.getChannel()
.size()];
                    fis.read(buf);
                    // *** POINT 2 *** Handle received result da
ta carefully and securely,
                    // even though the data came from in-house a
pplications.
                    // Omitted, since this is a sample. Please r
efer to "3.2 Handling Input Data Carefully and Securely."
```

```
                    logLine(new String(buf));
                } catch (IOException e) {
                    android.util.Log.e("InhouseUserActivity", "f
ailed to read file");
                } finally {
                    try {
                        fis.close();
                    } catch (IOException e) {
                        android.util.Log.e("ExternalFileActivity"
, "failed to close file");
                    }
                }
            }
            try {
                pfd.close();
            } catch (IOException e) {
                android.util.Log.e("ExternalFileActivity", "fail
ed to close file descriptor");
            }
        } else {
            logLine(" null file descriptor");
        }
    }

    private TextView mLogView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mLogView = (TextView) findViewById(R.id.logview);
    }

    private void logLine(String line) {
        mLogView.append(line);
        mLogView.append("¥n");
    }
}
```

4.6.3.2 为目录设置访问权限

以上所解释的安全考虑，重点在于文件。 还需要考虑作为文件容器的目录的安全性。 以下说明了目录的访问权限设置的安全性考虑。

在 Android 中，有一些方法可以在应用目录中获取/创建子目录。 主要如表 4.6-3。

表 4.6-3 在应用目录中获取/创建子目录的方法

	规定其它应用的访问权限	删除文件
`Context#getFilesDir()`	不可能（只有执行权限）	设置=>应用=>选择目标应用=>清除数据
`Context#getCacheDir()`	不可能（只有执行权限）	设置=>应用=>选择目标应用=>清除缓存（也可以清除数据）
`Context#getDir(String name,		

int
MODE)` | 可以对 MODE 设置如下： MODE_PRIVATE `MODE_WORLD_READABLE` `MODE_WORLD_WRITEABLE` | 设置=>应用=>选择目标应用=>清除数据 |

这里特别需要注意的是 `Context#getDir()` 的访问权限设置。 正如文件创建中所说明的，从安全设计的角度来看，目录基本上也应该设置为私有的。 当信息共享取决于访问权限设置时，可能会产生意想不到的副作用，所以应采取其他方法用于信息共享。

`MODE_WORLD_READABLE`

这是一个标志，为所有应用提供目录的只读权限。 所以所有应用都可以获取目录中的文件列表，和单个文件属性信息。 由于秘密文件可能不会被放置在这些目录中，所以通常不能使用该标志 [15]。

`MODE_WORLD_WRITEABLE`

该标志位其他应用提供目录的写入权限。 所有应用都可以创建/移动/重命名/删除目录中的文件。 这些操作与文件本身的访问权限设置（读/写/执行）没有关系，所以需要注意的是，仅仅使用目录的写入权限就能执行操作。 此标志允许其他应用随意删除或替换文件，因此一般不能使用。

> [15] `MODE_WORLD_READABLE` 和 `MODE_WORLD_WRITEABLE` 在 API 17 和更高版本以及 API 24 和更高版本中弃用，使用它们将触发安全异常。

对于表 4.6-3 “用户删除”，请参考“4.6.2.4 应用应考虑文件范围而设计（必需）”。

4.6.3.3 共享首选项和数据库文件的访问权限设置

共享首选项和数据库也由文件组成。 对于访问权限设置，对文件解释的内容也会在这里解释。 因此，共享首选项和数据库都应该创建为私有文件，与文件相同，内容共享应该由 Android 的应用间联动系统来实现。

下面将展示共享首选项的使用示例。 通过 `MODE_PRIVATE` ，共享首选项被设置为私有文件。

```
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;

// Ommision of a passage

// Get Shared Preference . (If there's no Shared Preference, it'
s to be created.)
// Point:Basically, specify MODE_PRIVATE mode.
SharedPreferences preference = getSharedPreferences(
    PREFERENCE_FILE_NAME, MODE_PRIVATE);

// Example of writing preference which value is charcter string
Editor editor = preference.edit();
editor.putString("prep_key", "prep_value");// key:"prep_key", va
lue:"prep_value"
editor.commit();
```

对于数据库，请参考“4.5 使用 SQLite”。

4.6.3.4 Android 4.4（API 级别 19）及更高版本中，外部存储访问的规范更改

自 Android 4.4（API Level 19）以来，外部存储访问的规范已更改为以下内容。

（1）如果应用需要读/写其外部存储器上的特定目录，则不需要使用 `<uses-permission>` 声明 `WRITE_EXTERNAL_STORAGE` / `READ_EXTERNAL_STORAGE` 权限。（已更改）

（2）如果应用需要读取除外部存储器上特定目录以外的目录中的文件，则需要使用 `<uses-permission>` 声明 `READ_EXTERNAL_STORAGE` 权限。 （已更改）

（3）如果应用需要写入主外部存储器上的特定目录以外的目录中的文件，则需要使用 `<uses-permission>` 声明 `WRITE_EXTERNAL_STORAGE` 权限。

（4）应用无法写入次要外部存储器上的特定目录以外的目录中的文件。

在该规范中，根据 Android OS 的版本确定是否需要权限请求。 因此，如果应用支持包括 Android 4.3 和 4.4 在内的版本，则可能会导致应用需要用户不必要的许可。 因此，建议使用对应（1）的应用，如下所示使用 `<uses-permission>` 的 `maxSdkVersion` 属性。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.file.externaluser" >
    <!-- In Android 4.0.3 (API Level 14) and later, the permissi
on for reading external storages
    has been defined and the application should decalre that it
requires the permission.
    In fact in Android 4.4 (API Level 19) and later, that must b
e declared to read other directories
    than the package specific directories. -->
    <uses-permission android:name="android.permission.READ_EXTER
NAL_STORAGE" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:allowBackup="false" >
        <activity
            android:name=".ExternalUserActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

4.6.3.5 Android 7.0（API Level 24）中的规范已修改，以便访问外部存储介质上的特定目录

在运行 Android 7.0（API Level 24）或更高版本的设备上，引入了一种称为作用域目录访问 API的新 API。 作用域目录访问允许应用在未经许可的情况下，访问外部存储器上的特定目录。 在作用域目录访问中，将 `Environment` 类中定义的目录作为参数传递给 `StorageVolume#createAccessIntent` 方法，来创建一个意图。 通过 `startActivityForResult` 发送此意图，可以启动一个对话框，在终端屏幕上请求访问权限，并且 - 如果用户授予权限 - 每个存储卷上的指定目录都可以访问。

表 4.6-4 可以通过作用域目录访问来访问的目录

DIRECTORY_MUSIC	通用音乐文件的标准位置
DIRECTORY_PODCASTS	播客的标准目录
DIRECTORY_RINGTONES	ringtone 的标准目录
DIRECTORY_ALARMS	闹铃的标准目录
DIRECTORY_NOTIFICATIONS	提醒的标准目录
DIRECTORY_PICTURES	图片的标准目录
DIRECTORY_MOVIES	电影的标准目录
DIRECTORY_DOWNLOADS	用户下载的文件的标准目录
DIRECTORY_DCIM	相机产生的图片/视频文件的标准目录
DIRECTORY_DOCUMENTS	用户创建的文档的标准目录

如果应用要访问的位置位于上述目录之一，并且该应用正在 Android 7.0 或更高版本的设备上运行，则建议使用作用域目录访问，原因如下。 对于必须继续支持 Android 7.0 以下的设备的应用，请参阅“4.6.3.4 Android 4.4（API级别19）及更高版本中的外部存储访问的规范更改”中，列出的 `AndroidManifest` 中的示例代码。

- 授予访问外部存储的权限时，应用可以访问预期目标以外的目录。
- 使用存储器访问框架来要求用户选择可访问的目录，会导致繁琐的过程，用户必须在每次访问时配置一个选择器。 另外，当访问外部存储器的根目录时，整个存储器变成可访问的。

4.7 使用可浏览的意图

Android 应用可以设计为从浏览器启动，并对应网页链接。 这个功能被称为“可浏览的意图”。 通过在清单文件中指定 URI 模式，应用将响应具有其 URI 模式的链接转移（用户点击等），并且应用以链接作为参数启动。

此外，使用 URI 模式从浏览器启动相应应用的方法不仅支持 Android，也支持 iOS 和其他平台，这通常用于 Web 应用与外部应用之间的链接等。例如， 在 Twitter 应用或 Facebook 应用中定义了以下 URI 模式，并且在 Android 和 iOS 中从浏览器启动相应的应用。

表 4.7-1

URL 模式	相应应用
`fb://`	Facebook
`twitter://`	Twitter

考虑到联动性和便利性，功能似乎非常方便，但存在一些风险，即该功能被恶意第三方滥用。 可以假设的是，它们滥用应用功能，通过准备一个恶意网站，它的链接的 URL 具有不正确的参数，或者它们通过欺骗智能手机用户安装恶意软件，它包含相同的 URI 模式，来获取包含在 URL 中的信息。 使用“可浏览的意图”来对付这些风险时有一些要注意的地方。

4.7.1 示例代码

使用“可浏览的意图”的应用的示例代码如下：

要点：

1) （网页侧）不得包含敏感信息。

2) 仔细和安全地处理 URL 参数。

Starter.html

```
<html>
    <body>
        <!-- *** POINT 1 *** Sensitive information must not be i
ncluded -->
        <!-- Character strings to be passed as URL parameter, sh
ould be UTF-8 and URI encoded. -->
        <a href="secure://jssec?user=user_id"> Login </a>
    </body>
</html>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:allowBackup="false" >
        <activity
            android:name=".BrowsableIntentActivity"
            android:label="@string/title_activity_browsable_inte
nt"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.VIEW"
 />
                // Accept implicit Intent
                <category android:name="android.intent.category.
DEFAULT" />
                // Accept Browsable intent
                <category android:name="android.intent.category.
BROWSABLE" />
                // Accept URI 'secure://jssec'
                <data android:scheme="secure" android:host="jsse
c"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

BrowsableIntentActivity.java

```
package org.jssec.android.browsableintent;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.widget.TextView;

public class BrowsableIntentActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_browsable_intent);
        Intent intent = getIntent();
        Uri uri = intent.getData();
        if (uri != null) {
            // Get UserID which is passed by URI parameter
            // *** POINT 2 *** Handle the URL parameter carefull
y and securely.
            // Omitted, since this is a sample. Please refer to
"3.2 Handling Input Data Carefully and Securely."
            String userID = "User ID = " + uri.getQueryParameter(
"user");
            TextView tv = (TextView)findViewById(R.id.text_useri
d);
            tv.setText(userID);
        }
    }
}
```

4.7.2 规则书

使用“可浏览的意图”时，需要遵循以下规则：

4.7.2.1 （网页端）敏感信息不得包含在相应链接的参数中（必需）

当点击浏览器中的链接时，会发出一个意图，该意图的数据中有 URL 值（可以通过 `Intent#getData` 获取），并且带有相应意图过滤器的应用，从 Android 系统启动。

此时，当几个应用设置意图过滤器来接收相同的 URI 模式时，应用选择对话框将显示，与隐式意图正常启动相同，并启动用户选择的应用。 如果应用选择对话框中列出了恶意软件，则用户可能会错误地启动恶意软件，并将 URL 中的参数发送到恶意软件。

如上所述，需要避免直接在 URL 参数中包含敏感信息，因为它用于创建一般网页链接，所有包含在网页链接 URL 中的参数都可以提供给恶意软件。

用户 ID 和密码包含在 URL 中的例子：

```
insecure://sample/login?userID=12345&password=abcdef
```

此外，即使 URL 参数仅包含非敏感内容，如用户ID，在由'可浏览的意图'启动后，在应用中输入密码时，用户可能会启动恶意软件并向其输入密码。所以应该考虑，一些规范，例如整个登录过程，在应用端完成。 在设计应用时必须记住它，并且由'可浏览的意图'启动应用，等同于由隐式意图启动，并且不保证启动了有效的应用。

4.7.2.2 小心和安全地处理 URL 参数（必需）

发送给应用的 URL 参数，并不总是来自合法的 Web 页面，因为匹配 URI 模式链接不仅可以由开发者生成，也可以由任何人生成。 另外，没有方法可以验证 URL 参数是否从有效网页发送。

因此，在使用 URL 参数之前，有必要验证 URL 参数的安全性，例如，检查是否包含意外值。

4.8 输出到 LogCat

在 Android 中有一种名为 LogCat 的日志机制，不仅系统日志信息，还有应用日志信息也会输出到 LogCat。 LogCat 中的日志信息可以从同一设备中的其他应用中读出 [17]，因此向L ogcat 输出敏感信息的应用，被认为具有信息泄露的漏洞。 敏感信息不应输出到 LogCat。

> [17] 输出到 LogCat 的日志信息，可以由声明 `READ_LOGS` 权限的应用读取。 但是，在 Android 4.1 及更高版本中，无法读取其他应用输出的日志信息。 但智能手机用户可以通过 ADB ，阅读输出到 logcat 的每个日志信息。

从安全角度来看，在发行版应用中，最好不要输出任何日志。 但是，即使在发行版应用的情况下，在某些情况下也会出于某种原因输出日志。 在本章中，我们将介绍一些方法，以安全的方式将消息输出到 LogCat，即使在发行版应用中也是如此。 除此解释外，请参考“4.8.3.1 发行版应用中日志输出的两种思路”。

4.8.1 示例代码

接下来是在发行版应用中，通过 ProGuard 控制输出到 LogCat 的日志的方法。 ProGuard 是自动删除不需要的代码（如未使用的方法等）的优化工具之一。

`android.util.Log` 类有五种类型的日志输出方法： `Log.e()` , `Log.w()` , `Log.i()` , `Log.d()` , `Log.v()` 。对于日志信息，有意输出的日志信息（以下称为“操作日志信息”）应该区分于不适合发行版应用的信息（以下称为开发日志信息），例如调试日志。建议使用 `Log.e()/w()/i()` 输出操作日志信息，并使用 `Log.d()/v()` 输出开发日志。正确使用五种日志输出方法的详细信息，请参阅“4.8.3.2 日志级别和日志输出方法的选择标准”，另外请参考“4.8.3.3 调试日志和 `VERBOSE` 日志并不总是自动删除”。

这是一个以安全方式使用 LogCat 的例子。此示例包括用于输出调试日志的 `Log.d()` 和 `Log.v()` 。如果应用用于发布，这两种方法将被自动删除。在此示例代码中，ProGuard 用于自动删除调用 `Log.d()/v()` 的代码块。

要点：

1) 敏感信息不能由 `Log.e()/w()/i()` ， `System.out / err` 输出。

2) 敏感信息应由 `Log.d()/v()` 在需要时输出。

3) 不应使用 `Log.d()/v()` 的返回值（以替换或比较为目的）。

4) 当你构建应用来发布时，你应该在代码中引入机制，自动删除不合适的日志记录方法（如 `Log.d()` 或 `Log.v()` ）。

5) 必须使用发行版构建配置来创建用于（发布）发行的 APK 文件。

ProGuardActivity.java

```java
package org.jssec.android.log.proguard;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

public class ProGuardActivity extends Activity {

    final static String LOG_TAG = "ProGuardActivity";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_proguard);
        // *** POINT 1 *** Sensitive information must not be out
put by Log.e()/w()/i(), System.out/err.
        Log.e(LOG_TAG, "Not sensitive information (ERROR)");
        Log.w(LOG_TAG, "Not sensitive information (WARN)");
        Log.i(LOG_TAG, "Not sensitive information (INFO)");
        // *** POINT 2 *** Sensitive information should be outpu
t by Log.d()/v() in case of need.
        // *** POINT 3 *** The return value of Log.d()/v()should
 not be used (with the purpose of substitution or comparison).
        Log.d(LOG_TAG, "sensitive information (DEBUG)");
        Log.v(LOG_TAG, "sensitive information (VERBOSE)");
    }
}
```

proguard-project.txt

```
# prevent from changing class name and method name etc.
-dontobfuscate
# *** POINT 4 *** In release build, the build configurations in
which Log.d()/v() are deleted automatica
lly should be constructed.
-assumenosideeffects class android.util.Log {
    public static int d(...);
    public static int v(...);
}
```

要点 5：必须使用发行版构建配置来创建用于（发布）发行的 APK 文件。

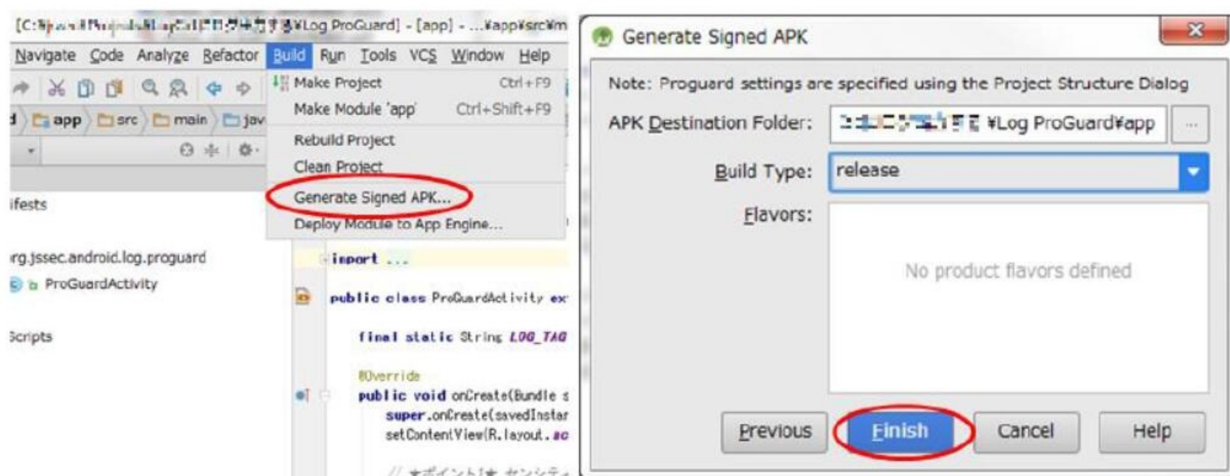


Figure 4.8-1 How to create release version application

开发版应用（调试版本）和发行版应用（发布版本）之间的LogCat 输出差异如下图 4.8-2 所示。

Development version application (Debug build)

LogCat | Search | Console

Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag

Level	Tag	Text
E	ProGuardActivity	Not sensitive information (ERROR)
W	ProGuardActivity	Not sensitive information (WARN)
I	ProGuardActivity	Not sensitive information (INFO)
D	ProGuardActivity	sensitive information (DEBUG)
V	ProGuardActivity	sensitive information (VERBOSE)

Release version application (Release build)

LogCat | Search | Console

Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag

Level	Tag	Text
E	ProGuardActivity	Not sensitive information (ERROR)
W	ProGuardActivity	Not sensitive information (WARN)
I	ProGuardActivity	Not sensitive information (INFO)

Figure 4.8-2 Difference of LogCat output
between development version application and release version application

4.8.2 规则书

输出消息记录时，遵循以下规则：

4.8.2.1 操作日志信息中不能包含敏感信息（必需）

输出到 LogCat 的日志可以从其他应用中读取，因此敏感信息（如用户的登录信息）不应该由发行版应用输出。 在开发过程中，不必编写输出敏感信息的代码，或者在发布之前需要删除所有这些代码。

为了遵循这个规则，首先，不要在操作日志信息中包含敏感信息。 此外，建议构建系统，在构建发行版时，删除输出敏感信息的代码。 请参阅“4.8.2.2 构建生成系统，在构建发行版时，自动删除输出开发日志信息的代码（推荐）”。

4.8.2.2 构建生成系统，在构建发行版时，自动删除输出开发日志信息的代码（推荐）

开发应用时，有时最好将敏感信息输出到日志中，来检查过程内容和调试，例如复杂逻辑过程中的临时操作结果，程序内部状态信息，通信协议的数据结构。在开发过程中，将敏感信息作为调试日志输出并不重要，在这种情况下，相应的日志输出代码应该在发布之前删除，如“4.8.2.1 操作日志信息中不能包含敏感信息（必需）”所述。

为了在构建发行版时，确实删除了输出开发日志信息的代码，应该构建系统，使用某些工具自动执行代码删除。“4.8.1 示例代码”中介绍的 ProGuard 可以用于此方法。如下所述，用 ProGuard 删除代码有一些值得注意的地方。这里应该将系统用于一些应用，它通过 `Log.d()/ v()` 输出开发日志信息，根据“4.8.3.2 日志级别和日志输出方法的选择标准”。

ProGuard 会自动删除不需要的代码，如未使用的方法。通过指定 `Log.d()/ v()` 作为 `-assumenosideeffects` 选项的参数， `Log.d()`， Log.v()`的调用被视为不必要的代码，并且这些代码将被删除。

```
-assumenosideeffects class android.util.Log {
    public static int d(...);
    public static int v(...);
}
```

如果使用这个自动删除系统，请注意 `Log.d()` ， `Log.v()` 代码在使用其返回值时不会被删除，因此不应该使用 `Log.d()` ， `Log.v()` 的返回值。 例如，下一个代码中的 `Log.v()` 不会被删除。

```
int i = android.util.Log.v("tag", "message");
System.out.println(String.format("Log.v() returned %d. ", i)); /
/Use the returned value of Log.v() for examination
```

如果你想重复使用源代码，则应保持项目环境的一致性，包括 ProGuard 设置。 例如，预设 `Log.d()` 和 `Log.v()` 的源代码将被上面的 ProGuard 设置自动删除。 如果在未设置 ProGuard 的其他项目中使用此源代码，则不会删除 `Log.d()` 和 `Log.v()` ，因此可能会泄露敏感信息。 重用源代码时，应确保包括 ProGuard 设置在内的项目环境的一致性。

4.8.2.3 输出 `Throwable` 对象时，使用 `Log.d()/v()` （推荐）

如“4.8.1 示例代码”和“4.8.3.2 日志级别和日志输出方法的选择标准”中所述，输出敏感信息不应通过 `Log.e()/w()/i()` 输出来记录。 另一方面，为了使开发者输出程序异常的细节来记录，当异常发生时，在某些情况下，堆栈踪迹通过 `Log.e(..., Throwable tr)/w(..., Throwable tr)/i(..., Throwable tr)` 输出到 LogCat。 但是，敏感信息有时可能包含在堆栈踪迹中，因为它显示程序的详细内部结构。 例如，当 `SQLiteException` 按原样输出时，会输出 SQL 语句的类型，因此可能会提供 SQL 注入攻击的线索。 因此，建议在输出 `Throwable` 对象时，仅使用 `Log.d()/v()` 方法。

4.8.2.4 仅仅将 `android.util.Log` 类的方法用于日志输出（推荐）

在开发过程中，你可以通过 `System.out / err` 输出日志，来验证应用的行为是否按预期工作。 当然，日志可以通过 `System.out / err` 的 `print()/ println()` 方法输出到 LogCat，但强烈建议仅使用 `android.util.Log` 类的方法，原因如下。

在输出日志时，一般根据信息的紧急程度，正确使用最合适的输出方法，并控制输出。 例如，使用严重错误，注意，简单应用的信息通知等类别。 然而，在这种情况下，在发布时需要输出的信息（操作日志信息），和可以包括敏感信息（开发日志信息）的信息，通过相同的方法输出。 所以，当删除输出敏感信息的代码时，可能会存在一些删除操作被忽略掉的危险。

除此之外，当使用 `android.util.Log` 和 `System.out / err` 进行日志输出时，与仅使用 `android.util.Log` 相比，需要考虑的因素会增加，因此可能会出现一些错误，比如 一些删除被忽略掉了。

为了减少上述错误发生的风险，建议仅使用 `android.util.Log` 类的方法。

4.8.3 高级话题

4.8.3.1 发布版应用中日志输出的两种思路

发布版应用中有两种思考日志输出的方式。一个是任何日志都不应该输出，另一个是用于以后分析的必要信息应该作为日志输出。从安全角度来看，最好是，任何日志都不应该在发行版应用中输出，但有时候，即使在发行版本应用中，出于各种原因也会输出日志。每种思考方式按照以下描述。

前者是“任何日志都不应该输出”，这是因为，在发行版应用中输出日志没有那么重要，并且存在泄露敏感信息的风险。这是因为开发人员没有办法在 Android 应用运行环境中收集发行版应用的日志信息，这与许多 Web 应用的运行环境不同。基于这种思想，日志代码仅用于开发阶段，并且在构建发行版应用时删除所有日志代码。

后者是“必要的信息应作为日志输出，以供日后分析”，作为客户支持中，分析应用错误的最终选项，以防你的客户支持有任何疑问。 基于这个想法，如上所述，有必要准备系统来防止人为错误并将其引入到项目中，因为如果你没有系统，则必须记住避免在发行版应用中记录敏感信息。

更多日志方法的信息，请参考下面的链接：

适用于贡献者/日志的代码风格指南

http://source.android.com/source/code-style.html#log-sparingly

4.8.3.2 日志级别和日志输出方法的选择标准

在 Android 中的 `android.util.Log` 类中定义了五个日志级别（ `ERROR` ， `WARN` ， `INFO` ， `DEBUG` ， `VERBOSE` ）。使用 `android.util.Log` 类输出日志消息时，应该选择最合适的方法，如表 4.8-1 所示，它展示了日志级别和方法的选择标准。

表 4.8-1 日志级别和方法的选择标准

日志级别	方法	要输出的日志信息
`ERROR`	`Log.e()`	应用处于错误状态时，输出的日志信息
`WARN`	`Log.w()`	应用面临非预期严重情况时，输出的日志信息
`INFO`	`Log.i()`	与上面不同，用于提示应用状态中任何值得注意的更改或者结果
`DEBUG`	`Log.d()`	应用的内部状态信息，开发应用时，需要临时输出，用于分析特定 bug 的成因
`VERBOSE`	`Log.v()`	不属于上面任何一个的日志信息。应用开发者以多种目的输出。例如，输出服务器通信信息来转储。

发行版应用的注意事项：

`e/w/i` ：

日志信息可能由用户参考，因此可以在开发版应用和发行版应用中输出。 因此，敏感信息不应该在这些级别输出。

`d/v` ：

日志信息仅适用于应用开发人员。 因此，这种类型的信息不应该在发行版的情况下输出。

更多日志方法的信息，请参考下面的链接：

适用于贡献者/日志的代码风格指南

http://source.android.com/source/code-style.html#log-sparingly

4.8.3.3 `DEBUG` 和 `VERBOSE` 日志并不总是自动删除

以下引用自 `android.util.Log` 类 [18] 的开发人员参考。

> [18] ttp://developer.android.com/reference/android/util/Log.html

按照啰嗦程度的顺序排列，从最少到最多是 `ERROR` ， `WARN` ， `INFO` ， `DEBUG` ， `VERBOSE` 。除了在开发期间，绝不应该将 `VERBOSE` 编译进应用。 `DEBUG` 日志被编译但在运行时剥离。 始终保留 `ERROR` ， `WARN` ， `INFO` 日志。

在阅读了上述文章之后，一些开发人员可能会误解 `Log` 类的行为，如下所示。

- 构建发行版时不编译 `Log.v()` 调用， `VERBOSE` 日志从不输出。
- 编译 `Log.v()` 调用，但执行时绝不输出 `DEBUG` 日志。

但是，日志记录方法从来不会表现成这样，并且无论使用调试模式还是发布模式编译，都会输出所有消息。 如果仔细阅读文档，你将能够认识到，文档的要点与日志方法的行为无关，而是日志的基本策略。

在本章中，我们通过使用 ProGuard 引入了示例代码以获得上述的预期结果。

4.8.3.4 从汇编中移除敏感信息

如果为了删除 `Log.d()` 方法而使用 ProGuard 构建以下代码，有必要记住， `ProGuard` 会保留为日志信息构造字符串的语句（代码的第一行），即使它删除了 `Log.d()` 方法的调用（代码的第二行）。

```
String debug_info = String.format("%s:%s", "Sensitive informatio
n1", "Sensitive information2");
if (BuildConfig.DEBUG) android.util.Log.d(TAG, debug_info);
```

以下反汇编显示了使用 ProGuard 发布上述代码的结果。 实际上，没有 `Log.d()` 调用过程，但你可以看到字符串一致性定义，例如 `Sensitive information1` ，和 `String#format()` 方法的调用过程，不会被删除并仍然存在。

```
const-string v1, "%s:%s"
const/4 v2, 0x2
new-array v2, v2, [Ljava/lang/Object;
const/4 v3, 0x0
const-string v4, "Sensitive information 1"
aput-object v4, v2, v3
const/4 v3, 0x1
const-string v4, "Sensitive information 2"
aput-object v4, v2, v3
invoke-static {v1, v2}, Ljava/lang/String;->format(Ljava/lang/St
ring;[Ljava/lang/Object;)Ljava/lang
/String;
move-result-object v0
```

实际上，找到反汇编 APK 文件的组装日志输出信息特定部分并不容易。 但是，在某些处理机密信息的应用中，这种类型的过程在某些情况下不应保留在 APK 文件中。 你应该像下面那样实现你的应用，来避免在字节码中保留敏感信息的后果。在发行版中，编译器优化将完全删除以下代码。

```
if (BuildConfig.DEBUG) {
String debug_info = String.format("%s:%s", " Snsitive informatio
n 1", "Sensitive information 2");
if (BuildConfig.DEBUG) android.util.Log.d(TAG, debug_info);
}
```

此外，ProGuard 无法删除以下日志消息代码 `("result:" + value)` 。

```
Log.d(TAG, "result:" + value);
```

在这种情况下，你可以通过以下方式解决问题。

```
if (BuildConfig.DEBUG) Log.d(TAG, "result:" + value);
```

4.8.3.5 意图的内容输出到了 LogCat

使用活动时，需要注意，因为 `ActivityManager` 将意图的内容输出到 LogCat。请参阅“4.1.3.5 使用活动时的日志输出”。

4.8.3.6 限制输出到 `System.out / err` 的日志

`System.out / err` 方法将所有消息输出到 LogCat。 即使开发者没有在他们的代码中使用这些方法，Android 也可以向 `System.out / err` 发送一些消息，例如，在以下情况下，Android 会将堆栈踪迹发送到 `System.err` 方法。

- 使用 `Exception#printStackTrace()` 时
- 隐式输出到 `System.err` 时（当异常没有被应用捕获时，它会由系统提供给 `Exception#printStackTrace()` 。）

你应该适当地处理错误和异常，因为堆栈踪迹包含应用的独特信息。

我们介绍一种改变 `System.out / err` 默认输出目标的方法。 当你构建发行版应用时，以下代码将 `System.out / err` 方法的输出重定向到任何地方。 但是，你应该考虑此重定向是否会导致应用或系统故障，因为代码会暂时覆盖 `System.out / err` 方法的默认行为。 此外，这种重定向仅对你的应用有效，对系统进程毫无价值。

OutputRedirectApplication.java

```
package org.jssec.android.log.outputredirection;

import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintStream;
import android.app.Application;

public class OutputRedirectApplication extends Application {

    // PrintStream which is not output anywhere
    private final PrintStream emptyStream = new PrintStream(new
OutputStream() {
        public void write(int oneByte) throws IOException {
            // do nothing
        }
    });

    @Override
    public void onCreate() {
        // Redirect System.out/err to PrintStream which doesn't
output anywhere, when release build.
        // Save original stream of System.out/err
        PrintStream savedOut = System.out;
        PrintStream savedErr = System.err;
        // Once, redirect System.out/err to PrintStream which do
esn't output anywhere
        System.setOut(emptyStream);
        System.setErr(emptyStream);
        // Restore the original stream only when debugging. (In
release build, the following 1 line is deleted byProGuard.)
        resetStreams(savedOut, savedErr);
    }

    // All of the following methods are deleted byProGuard when
release.
    private void resetStreams(PrintStream savedOut, PrintStream
savedErr) {
        System.setOut(savedOut);
        System.setErr(savedErr);
    }
}
```

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.log.outputredirection" >
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:name=".OutputRedirectApplication"
        android:allowBackup="false" >
        <activity
            android:name=".LogActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

proguard-project.txt

```
# Prevent from changing class name and method name, etc
-dontobfuscate
# In release build, delete call from Log.d()/v() automatically.
-assumenosideeffects class android.util.Log {
    public static int d(...);
    public static int v(...);
}
# In release build, delete resetStreams() automatically.
-assumenosideeffects class org.jssec.android.log.outputredirecti
on.OutputRedirectApplication {
    private void resetStreams(...);
}
```

开发版应用（调试版）和发布版应用（发行版）之间的 LogCat 输出差异如下图 4.8-3 所示。

Development version application (Debug build)

LogCat | Search | Console

Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag

Level	Tag	Text
I	LogActivity	Output logs by Log.i() (1st time)
I	System.out	output logs to System.out
W	System.err	output logs to System.err
I	LogActivity	Output logs by Log.i() (2nd time)

Release version application (Release build)

LogCat | Search | Console

Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag

Level	Tag	Text
I	LogActivity	Output logs by Log.i() (1st time)
I	LogActivity	Output logs by Log.i() (2nd time)

Figure 4.8-3 Difference of System.out/err in LogCat output, between development application and release application

4.9 使用 `WebView`

`WebView` 使你的应用能够集成 HTML / JavaScript 内容。

4.9.1 示例代码

我们需要采取适当的行动，具体取决于我们想通过 `WebView` 展示的内容，尽管我们可以通过它轻松展示网站和 html 文件。 而且我们还需要考虑来自 `WebView` 卓越功能的风险; 如 JavaScript-Java 对象绑定。 我们特别需要关注 JavaScript。（请注意 JavaScript 默认是禁用的，我们可以通过 `WebSettings#setJavaScriptEnabled()` 来启用它。 启用 JavaScript 存在潜在的风险，即恶意第三方可以获取设备信息并操作设备。 以下是使用 `WebView` [19] 的应用的原则：

> [19] 严格地说，如果我们可以说内容是安全的，你可以启用 JavaScript。 如果内容是在内部管理的，则内容应该保证安全。 公司可以保护他们。 换句话说，我们需要让企业代表的决策，来为其他公司的内容启用 JavaScript。 由可信伙伴开发的内容可能会有安全保证。 但仍有潜在风险。 因此，负责人需要作出决定。

（1）如果应用使用内部管理的内容，则可以启用 JavaScript。

（2）除上述情况外，你不应启用 JavaScript。

图 4.9-1 显示了根据内容特征选择示例代码的流程图。

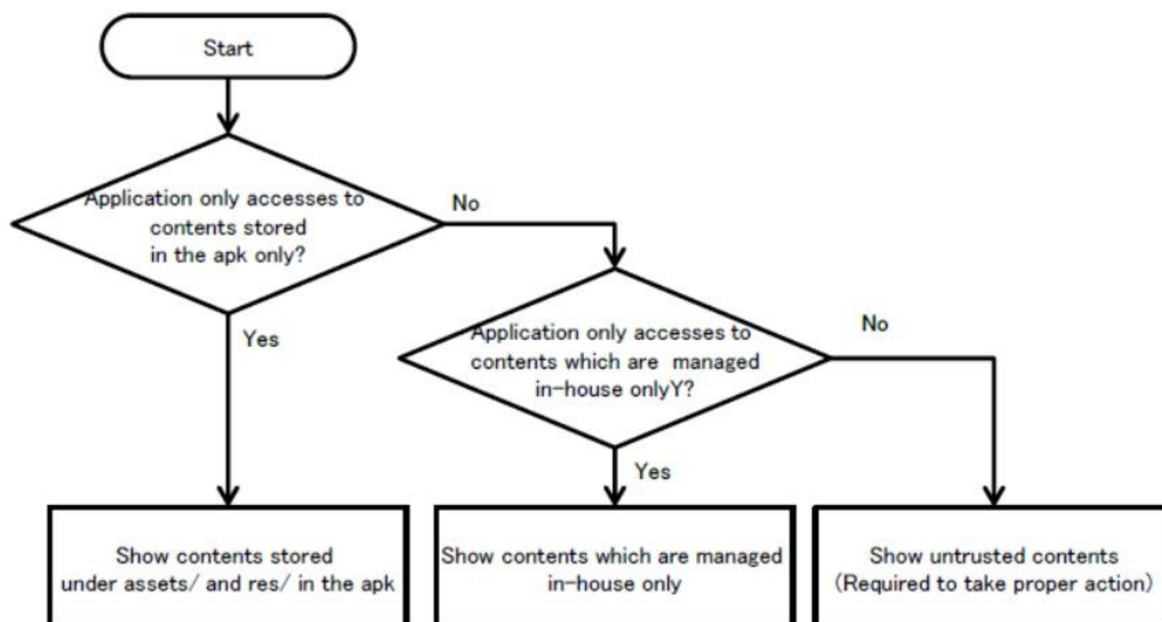


Figure 4.9-1 Flow Figure to select Sample code of WebView

4.9.1.1 仅显示存储在 APK 中的 `assets / res` 目录下的内容

如果你的应用仅显示存储在 apk 中 `assets/` 和 `res/` 目录下的内容，则可以启用 JavaScript。

以下示例代码展示了，如何使用 `WebView` 显示存储在 `assets/` 和 `res/` 下的内容。

要点：

1) 禁止访问文件（apk 文件中的 `assets/` 和 `res/` 下的文件除外）。

2) 你可以启用 JavaScript。

WebViewAssetsActivity.java

```
package org.jssec.webview.assets;

import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;

public class WebViewAssetsActivity extends Activity {

    /**
     * Show contents in assets
     */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        WebView webView = (WebView) findViewById(R.id.webView);
        WebSettings webSettings = webView.getSettings();
        // *** POINT 1 *** Disable to access files (except files
 under assets/ and res/ in this apk)
        webSettings.setAllowFileAccess(false);
        // *** POINT 2 *** Enable JavaScript (Optional)
        webSettings.setJavaScriptEnabled(true);
        // Show contents which were stored under assets/ in this
 apk
        webView.loadUrl("file:///android_asset/sample/index.html"
);
    }
}
```

4.9.1.2 仅显示内部管理的内容

只有当你的网络服务和你的 Android 应用可以采取适当措施来保护它们时，你才可以启用 JavaScript 来仅仅显示内部管理的内容。

Web 服务端操作：如图 4.9-2 所示，你的 Web 服务只能引用内部管理的内容。 另外，Web 服务需要采取适当的安全措施。 因为你的网络服务涉及的内容可能存在风险，因此存在潜在风险；如恶意攻击代码注入，数据操作等。请参阅“4.9.2.1 仅在内容由内部管理时启用 JavaScript（必需）”。

Android 应用端操作：使用 HTTPS，应用只有在证书可信的情况下，才应与受管理的 Web 服务建立网络连接。

以下示例代码是一个活动，展示了内部管理的内容。

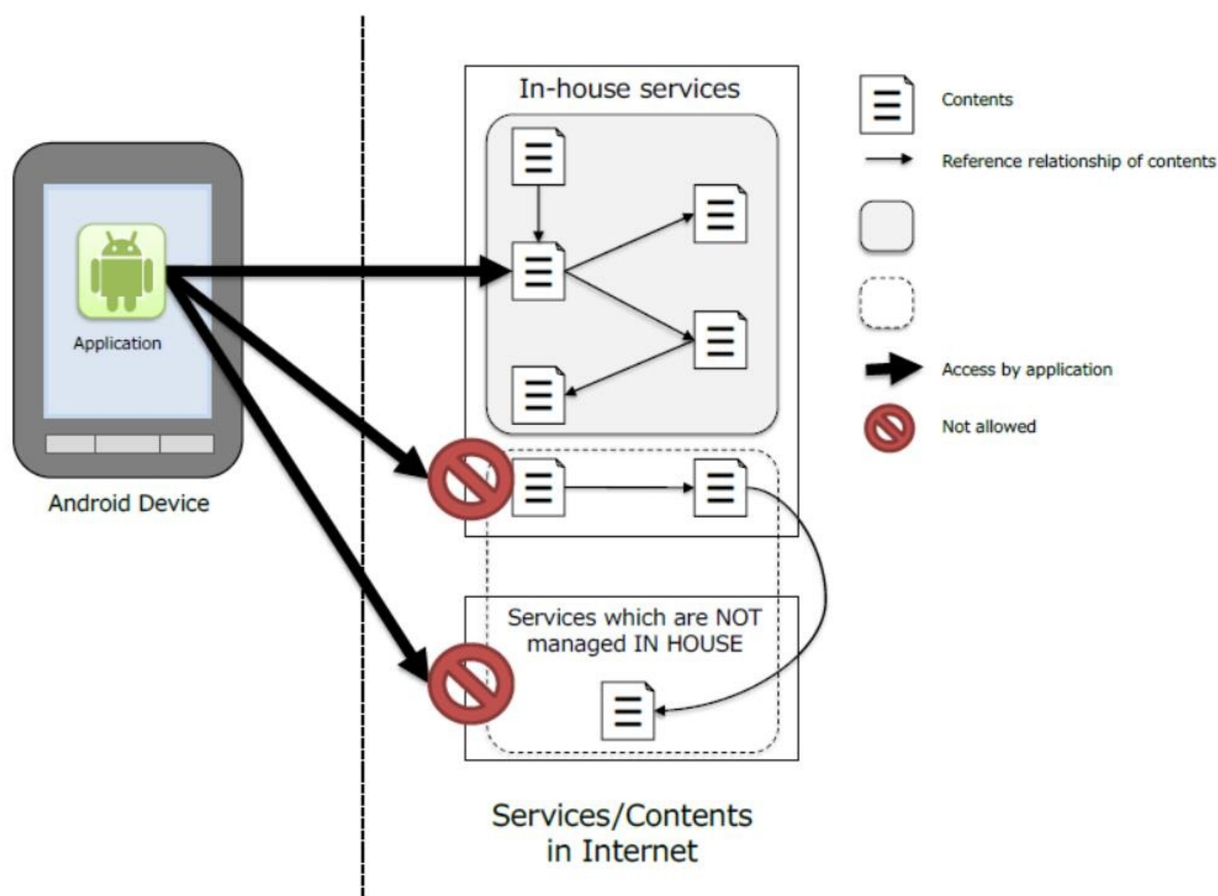


Figure 4.9-2 Accessible contents and Non-accessible contents from application

要点：

1) 适当处理来自 `WebView` 的 SSL 错误。

2) （可选）启用 `WebView` 的 JavaScript。

3) 将 URL 限制为 HTTPS 协议。

4) 将 URL 限制在内部。

WebViewTrustedContentsActivity.java

```
package org.jssec.webview.trustedcontents;

import android.app.Activity;
```

```
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.net.http.SslCertificate;
import android.net.http.SslError;
import android.os.Bundle;
import android.webkit.SslErrorHandler;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import java.text.SimpleDateFormat;

public class WebViewTrustedContentsActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        WebView webView = (WebView) findViewById(R.id.webView);
        webView.setWebViewClient(new WebViewClient() {
            @Override
            public void onReceivedSslError(WebView view,
                SslErrorHandler handler, SslError error) {
                // *** POINT 1 *** Handle SSL error from WebView
 appropriately
                // Show SSL error dialog.
                AlertDialog dialog = createSslErrorDialog(error)
;
                dialog.show();
                // *** POINT 1 *** Handle SSL error from WebView
 appropriately
                // Abort connection in case of SSL error
                // Since, there may be some defects in a certifi
cate like expiration of validity,
                // or it may be man-in-the-middle attack.
                handler.cancel();
            }
        });
        // *** POINT 2 *** Enable JavaScript (optional)
        // in case to show contents which are managed in house.
        webView.getSettings().setJavaScriptEnabled(true);
        // *** POINT 3 *** Restrict URLs to HTTPS protocol only
        // *** POINT 4 *** Restrict URLs to in-house
        webView.loadUrl("https://url.to.your.contents/");
    }

    private AlertDialog createSslErrorDialog(SslError error) {
        // Error message to show in this dialog
        String errorMsg = createErrorMessage(error);
        // Handler for OK button
        DialogInterface.OnClickListener onClickOk = new DialogIn
terface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int whic
h) {
```

```
                setResult(RESULT_OK);
            }
        };
        // Create a dialog
        AlertDialog dialog = new AlertDialog.Builder(
        WebViewTrustedContentsActivity.this).setTitle("SSL conne
ction error")
            .setMessage(errorMsg).setPositiveButton("OK", onClic
kOk)
            .create();
        return dialog;
    }

    private String createErrorMessage(SslError error) {
        SslCertificate cert = error.getCertificate();
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy
/MM/dd HH:mm:ss");
        StringBuilder result = new StringBuilder()
            .append("The site's certification is NOT valid. Conn
ection was disconnected.¥n¥nError:¥n");
        switch (error.getPrimaryError()) {
            case SslError.SSL_EXPIRED:
                result.append("The certificate is no longer vali
d.¥n¥nThe expiration date is ")
                    .append(dateFormat.format(cert.getValidNotAf
terDate()));
                return result.toString();
            case SslError.SSL_IDMISMATCH:
                result.append("Host name doesn't match. ¥n¥nCN="
)
                    .append(cert.getIssuedTo().getCName());
                return result.toString();
            case SslError.SSL_NOTYETVALID:
                result.append("The certificate isn't valid yet.¥
n¥nIt will be valid from ")
                    .append(dateFormat.format(cert.getValidNotBe
foreDate()));
                return result.toString();
            case SslError.SSL_UNTRUSTED:
                result.append("Certificate Authority which issue
d the certificate is not reliable.¥n¥nCertificate Authority¥n")
                    .append(cert.getIssuedBy().getDName());
                return result.toString();
            default:
                result.append("Unknown error occured. ");
                return result.toString();
        }
    }
}
```

4.9.1.3 显示非内部管理的内容

如果你的应用显示的内容不在内部管理，请勿启用 JavaScript，因为存在访问恶意内容的潜在风险。

以下示例代码是显示非内部管理的内容的活动。

此示例代码显示由用户通过地址栏输入的 URL 指定的内容。 请注意，当 JavaScript 错误发生时，JavaScript 被禁用并且连接中止。 HTTPS 通信的详细信息，错误处理与“4.9.1.2 仅显示内部管理的内容”相同。 详细信息请参阅“5.4 通过 HTTPS 进行通信”。

要点：

1) 适当处理来自 `WebView` 的 SSL 错误。

2) 禁用 `WebView` 的 JavaScript。

WebViewUntrustActivity.java

```
package org.jssec.webview.untrust;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.graphics.Bitmap;
import android.net.http.SslCertificate;
import android.net.http.SslError;
import android.os.Bundle;
import android.view.View;
import android.webkit.SslErrorHandler;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.Button;
import android.widget.EditText;
import java.text.SimpleDateFormat;

public class WebViewUntrustActivity extends Activity {

    /*
    * Show contents which are NOT managed in-house (Sample progr
am works as a simple browser)
    */
    private EditText textUrl;
    private Button buttonGo;
    private WebView webView;

    // Activity definition to handle any URL request
    private class WebViewUnlimitedClient extends WebViewClient {
        @Override
        public boolean shouldOverrideUrlLoading(WebView webView,
 String url) {
            webView.loadUrl(url);
            textUrl.setText(url);
            return true;
```

```
        }

        // Start reading Web page
        @Override
        public void onPageStarted(WebView webview, String url, B
itmap favicon) {
            buttonGo.setEnabled(false);
            textUrl.setText(url);
        }

        // Show SSL error dialog
        // And abort connection.
        @Override
        public void onReceivedSslError(WebView webview,
        SslErrorHandler handler, SslError error) {
            // *** POINT 1 *** Handle SSL error from WebView app
ropriately
            AlertDialog errorDialog = createSslErrorDialog(error
);
            errorDialog.show();
            handler.cancel();
            textUrl.setText(webview.getUrl());
            buttonGo.setEnabled(true);
        }

        // After loading Web page, show the URL in EditText.
        @Override
        public void onPageFinished(WebView webview, String url)
{
            textUrl.setText(url);
            buttonGo.setEnabled(true);
        }
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        webView = (WebView) findViewById(R.id.webview);
        webView.setWebViewClient(new WebViewUnlimitedClient());
        // *** POINT 2 *** Disable JavaScript of WebView
        // Explicitly disable JavaScript even though it is disab
led by default.
        webView.getSettings().setJavaScriptEnabled(false);
        webView.loadUrl(getString(R.string.texturl));
        textUrl = (EditText) findViewById(R.id.texturl);
        buttonGo = (Button) findViewById(R.id.go);
    }

    public void onClickButtonGo(View v) {
        webView.loadUrl(textUrl.getText().toString());
    }
```

```
    private AlertDialog createSslErrorDialog(SslError error) {
        // Error message to show in this dialog
        String errorMsg = createErrorMessage(error);
        // Handler for OK button
        DialogInterface.OnClickListener onClickOk = new DialogIn
terface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int whic
h) {
                setResult(RESULT_OK);
            }
        };
        // Create a dialog
        AlertDialog dialog = new AlertDialog.Builder(
            WebViewUntrustActivity.this).setTitle("SSL connectio
n error")
            .setMessage(errorMsg).setPositiveButton("OK", onClic
kOk)
            .create();
        return dialog;
    }

    private String createErrorMessage(SslError error) {
        SslCertificate cert = error.getCertificate();
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy
/MM/dd HH:mm:ss");
        StringBuilder result = new StringBuilder()
            .append("The site's certification is NOT valid. Conn
ection was disconnected.¥n¥nError:¥n");
        switch (error.getPrimaryError()) {
            case SslError.SSL_EXPIRED:
                result.append("The certificate is no longer vali
d.¥n¥nThe expiration date is ")
                    .append(dateFormat.format(cert.getValidNotAf
terDate()));
                return result.toString();
            case SslError.SSL_IDMISMATCH:
                result.append("Host name doesn't match. ¥n¥nCN="
)
                    .append(cert.getIssuedTo().getCName());
                return result.toString();
            case SslError.SSL_NOTYETVALID:
                result.append("The certificate isn't valid yet.¥
n¥nIt will be valid from ")
                    .append(dateFormat.format(cert.getValidNotBe
foreDate()));
                return result.toString();
            case SslError.SSL_UNTRUSTED:
                result.append("Certificate Authority which issue
d the certificate is not reliable.¥n¥nCertificate Authority¥n")
                    .append(cert.getIssuedBy().getDName());
                return result.toString();
            default:
```

```
                    result.append("Unknown error occured. ");
                    return result.toString();
                }
            }
        }
```

4.9.2 规则书

当你需要时候 `WebView` 时，遵循下列规则：

4.9.2.1 只在内容由内部管理时启用 JavaScript（必需）

对于 `WebView` 我们需要关注的是是否启用 JavaScript。 原则上，只有当应用访问内部管理的服务时，我们才能启用 JavaScript。 如果有可能访问非内部管理的服务，则不得启用 JavaScript。

内部管理的服务

如果应用访问内部开发的内容，并通过内部管理的服务器分发，我们可以说这些内容仅由贵公司修改。 另外，每个内容还需要仅仅引用存储在服务器中的内容，它们具有适当安全性。

在这种情况下，我们可以在 `WebView` 上启用 JavaScript。 请参阅“4.9.1.2 仅显示内部管理的内容”。

如果你的应用仅显示存储在 apk 中 `assets/` 和 `res/` 目录下的内容，你也可以启用 JavaScript。 请参阅“4.9.1.1 仅显示存储在 `assets / res` 目录下的内容”。

非内部管理的服务

你绝不能认为，你可以确保非内部管理的内容的安全性。 因此你必须禁用 JavaScript。 请参阅“4.9.1.3 显示非内部管理的内容”。

另外，如果内容存储在外部存储介质中，如 microSD，则必须禁用 JavaScript；因为其他应用可以修改内容。

4.9.2.2 使用 HTTPS 与内部管理的服务器进行通信（必需）

你必须使用 HTTPS 与内部管理的服务器通信，因为存在恶意第三方欺骗服务的潜在风险。

请参阅“4.9.2.4 正确处理 SSL 错误（必需）”和“5.4 通过 HTTPS 通信”。

4.9.2.3 禁用 JavaScript 来显示通过意图接收的 URL（必需）

如果你的应用需要显示从其他应用，以意图等形式传递的 URL，则不要启用 JavaScript。因为存在用恶意 JavaScript 显示恶意网页的潜在风险。

“4.9.1.2 仅显示内部管理的内容”部分中的示例代码，使用固定值 URL 显示内部管理的内容来确保安全。

如果你需要显示从意图收到的 URL，则必须确认该 URL 在内部管理的 URL 中。简而言之，应用必须使用正则表达式等白名单来检查 URL。另外，它应该是 HTTPS。

4.9.2.4 适当处理 SSL 错误（必需）

当 HTTPS 通信发生 SSL 错误时，你必须终止网络通信并通知用户错误。

SSL 错误显示了无效的服务器认证风险或 MTIM（中间人攻击）风险。 请注意， `WebView` 没有 SSL 错误的错误通知机制。 因此，你的应用必须显示错误通知，来向用户通知风险。 请参阅“4.9.1.2 仅显示内部管理的内容”和“4.9.1.3 显示非内部管理的内容”一节中的示例代码。

另外，你的应用必须终止带有错误通知的通信。 换句话说，你不可以这样做。

- 忽略错误来与服务保持通信。
- 重试 HTTP 通信而不是 HTTPS。

请参阅“5.4 通过 HTTPS 进行通信”中所述的详细信息。

`WebView` 的默认行为是，发生 SSL 错误时终止通信。 因此，我们需要添加显示 SSL 错误通知。 然后我们可以正确处理 SSL 错误。

4.9.3 高级话题

4.9.3.1 Android 4.1 或更低版本中由 `addJavascriptInterface()` 引起的漏洞

4.2（API Level 17）版本以下的 Android，具有由 `addJavascriptInterface()` 引起的漏洞，这可能允许攻击者通过 `WebView` 上的 JavaScript 调用 Android 本地方法（Java）。

正如“4.9.2.1 只在内容由内部管理时启用 JavaScript”中所述，如果服务可以访问内部控制之外的服务，则不得启用 JavaScript。

在 Android 4.2（API Level 17）或更高版本中，已采取措施，将漏洞限制为在 Java 源代码上使用 `@JavascriptInterface` 注释的方法，而不是所有注入的 Java 对象的方法。 但是，如果服务可以访问内部控制之外的服务，则必须禁用 JavaScript，像“4.9.2.1”中提到的那样。

4.9.3.2 由文件模式导致的问题

如果使用默认设置的 `WebView` ，应用具有访问权限的所有文件，都可以通过在网页中通过文件模式访问，而无论页面的来源如何。 例如，恶意网页可以通过使用文件模式，向应用的私有文件的 URI 发送请求，来访问存储在应用私有目录中的文

件。

如果服务可以访问内部控制之外的服务，则禁用 JavaScript 的方法如“4.9.2.1 只在内容由内部管理时启用 JavaScript（必需）”中所述。 这样做是为了防止发送恶意文件模式请求。

同样在 Android 4.1（API Level 16）或更高版本的情况下，可以使用 `setAllowFileAccessFromFileURLs()` 和 `setAllowUniversalAccessFromFil` 来限制通过文件模式的访问。

禁用文件模式

```
webView = (WebView) findViewById(R.id.webview);
webView.setWebViewClient(new WebViewUnlimitedClient());
WebSettings settings = webView.getSettings();
settings.setAllowUniversalAccessFromFileURLs(false);
settings.setAllowFileAccessFromFileURLs(false);
```

4.9.3.3 使用 Web 消息时指定发送者的来源

Android 6.0（API Level 23）增加了一个 API，用于实现 HTML5 Web 消息传送。Web 消息传送是一种在 HTML5 中定义的框架，用于在不同的浏览上下文之间，发送和接收数据 [20]。添加到 `WebView` 类的 `postWebMessage()` 方法是一种方法，通过 Web 消息传送定义的跨域消息传送协议处理数据传输。

> [20] http://www.w3.org/TR/webmessaging/

此方法从 `WebView` 已读入的浏览上下文中发送一个消息，该消息由其第一个参数指定; 然而，在这种情况下，有必要指定发送者的来源作为第二个参数。 如果指定的来源 [21] 与发送者上下文中的来源不符，则不会发送该消息。 通过以这种方式限制发送者来源，此机制旨在防止消息传递给非预期发送者。

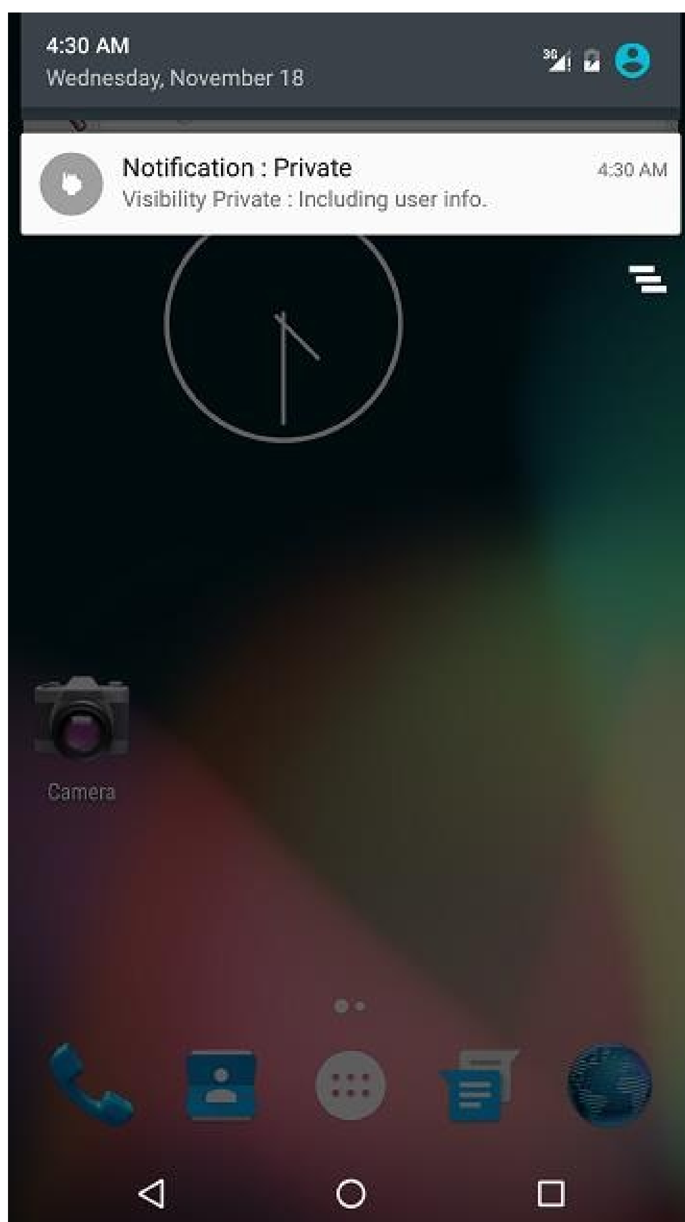
> [21] “来源”是一个 URL 模式以及一个主机名和端口号。 详细定义请参阅 http://tools.ietf.org/html/rfc6454。

但是，重要的是要注意，通配符可能被指定为 `postWebMessage()` 方法中的来源 [22]。如果指定了通配符，则不会检查消息的发送者来源，并且可以从任意来源发送消息。 在恶意内容已被读入 `WebView` 的情况下，如果发送重要消息时没有来源限制，则可能导致各种类型的损害。 因此，在使用 `WebView` 进行 Web 消息传递时，最好在 `postWebMessage()` 方法中明确指定特定的源。

> [22] 请注意，通配符是 `Uri.EMPTY` 和 `Uri.parse("")`（在编写 2016 年 9 月 1 日的版本时）。

4.10 使用通知

Android 提供用于向最终用户发送消息的通知功能。 使用通知会使一个称为状态栏的区域出现在屏幕上，你可以在其中显示图标和消息。



在 Android 5.0（API Level 21）中增强了通知的通信功能，即使在屏幕锁定时也可以通过通知显示消息，具体取决于用户和应用设置。 但是，不正确地使用通知，会导致私人信息（只应向最终用户自己显示）可能会被第三方看到。 出于这个原因，必须谨慎地注意隐私和安全性来实现此功能。

下表中总结了可见性选项的可能值和通知的相应行为。

可见性的值	通知行为
公共	通知会显示在所有锁定屏幕上
私有	通知显示在所有锁定的屏幕上；然而，在被密码保护的锁定屏幕上（安全锁），通知的标题和文本等字段是隐藏的（由公开可释放消息取代，私有信息是隐藏的）
秘密	通知不会显示在受密码或其他安全措施（安全锁）保护的锁定屏幕上。（通知显示在不涉及安全锁的锁定屏幕上。）

4.10.1 示例代码

当通知包含有关最终用户的私人信息时，必须从中排除了私人信息，之后才能添加到锁定屏幕来显示。

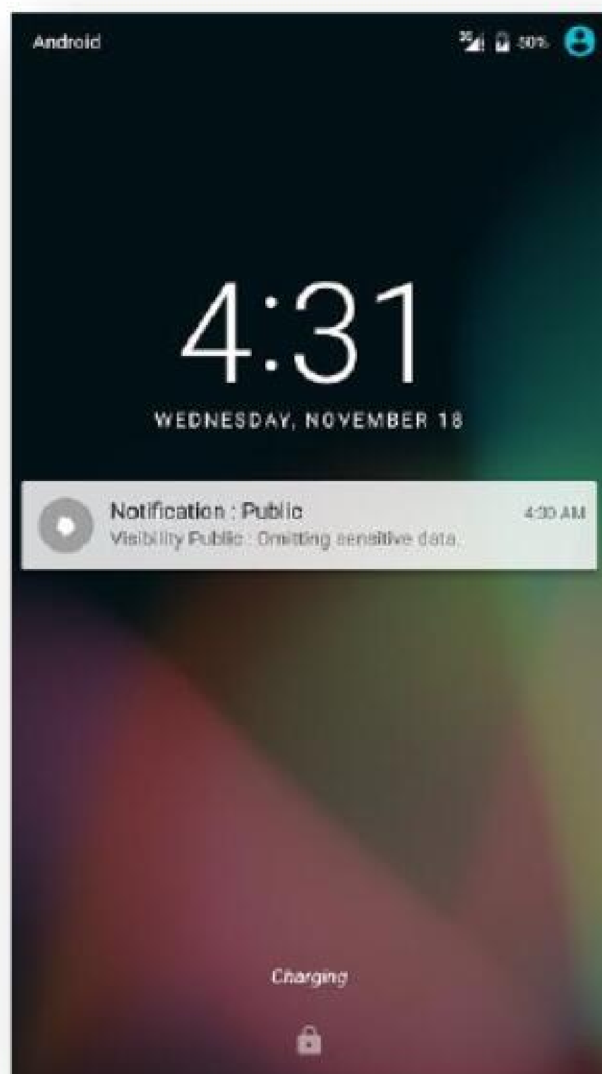


Figure 4.10-2 A notification on a locked screen

下面展示了示例代码，说明了如何正确将通知用于包含私人数据的消息。

要点：

1) 将通知用于包含私人数据的消息，请准备适合公开显示的通知版本（屏幕锁定时显示）。

2) 不要在公开显示的通知中包含隐私信息（屏幕锁定时显示）。

3) 创建通知时将可见性显示设置为私有。

4) 当可见性设置为私有时，通知可能包含私人信息。

VisibilityPrivateNotificationActivity.java

```
package org.jssec.notification.visibilityPrivate;
```

```
import android.app.Activity;
import android.app.Notification;
import android.app.NotificationManager;
import android.content.Context;
import android.os.Build;
import android.os.Bundle;
import android.view.View;

public class VisibilityPrivateNotificationActivity extends Activ
ity {

    /**
    * Display a private Notification
    */
    private final int mNotificationId = 0;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void onSendNotificationClick(View view) {
        // *** POINT 1 *** When preparing a Notification that in
cludes private information, prepare an additional Noficiation fo
r public display (displayed when the screen is locked).
        Notification.Builder publicNotificationBuilder = new Not
ification.Builder(this).setContentTitle("Notif
        ication : Public");
        if (Build.VERSION.SDK_INT >= 21)
            publicNotificationBuilder.setVisibility(Notification
.VISIBILITY_PUBLIC);
        // *** POINT 2 *** Do not include private information in
 Notifications prepared for public display (displayed when the s
creen is locked).
        publicNotificationBuilder.setContentText("Visibility Pub
lic : Omitting sensitive data.");
        publicNotificationBuilder.setSmallIcon(R.drawable.ic_lau
ncher);
        Notification publicNotification = publicNotificationBuil
der.build();
        // Construct a Notification that includes private inform
ation.
        Notification.Builder privateNotificationBuilder = new No
tification.Builder(this).setContentTitle("Notification : Private
");
        // *** POINT 3 *** Explicitly set Visibility to Private
when creating Notifications.
        if (Build.VERSION.SDK_INT >= 21)
            privateNotificationBuilder.setVisibility(Notificatio
n.VISIBILITY_PRIVATE);

        // *** POINT 4 *** When Visibility is set to Private, No
tifications may contain private information.
```

```
        privateNotificationBuilder.setContentText("Visibility Pr
ivate : Including user info.");
        privateNotificationBuilder.setSmallIcon(R.drawable.ic_la
uncher);
        // When creating a Notification with Visibility=Private,
 we also create and register a separate
        Notification with Visibility=Public for public display.
        if (Build.VERSION.SDK_INT >= 21)
            privateNotificationBuilder.setPublicVersion(publicNo
tification);
        Notification privateNotification = privateNotificationBu
ilder.build();
        //Although not implemented in this sample code, in many
cases
        //Notifications will use setContentIntent(PendingIntent
intent)
        //to ensure that an Intent is transmission when Notifica
tion
        //is clicked. In this case, it is necessary to take step
s--depending
        //on the type of component being called--to ensure that
the Intent
        //in question is called by safe methods (for example, by
 explicitly
        //using Intent). For information on safe methods for cal
ling various
        //types of component, see the following sections.
        //4.1. Creating and using Activities
        //4.2. Sending and receiving Broadcasts
        //4.4. Creating and using Services
        NotificationManager notificationManager = (NotificationM
anager) this.getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.notify(mNotificationId, privateNotif
ication);
    }
}
```

4.10.2 规则书

创建通知时，应该遵循下列规则：

4.10.2.1 无论可见性设置如何，通知都不得包含敏感信息（尽管私有信息是例外情况）（必需）

在使用 Android 4.3（API 级别 18）或更高版本的终端上，用户可以使用“设置”窗口，授予应用读取通知的权限。 获得此权限的应用将能够读取通知中的所有信息；因此，通知中不得包含敏感信息。 （但是，根据“可见性”设置，通知中可能会包含私有信息）。

通知中包含的信息通常不会被发送通知的应用以外的应用读取。 但是，用户可以明确将权限授予某些用户选择的应用，来读取通知中的所有信息。 因为只有用户已授予权限的应用才能读取通知中的信息，所以在通知中包含用户的私有信息没有任何问题。 另一方面，如果在通知中包括除了用户的私有信息之外的敏感信息（例如，仅由应用开发者知道的秘密信息），则用户自己可以尝试读取通知中包含的信息，并且可以授予应用权限来查看这些信息；因此包含私有用户信息以外的敏感信息是有问题的。

特定方法和条件请见“4.10.3.1 用户授予的查看通知的权限”。

4.10.2.2 可见性为公共的通知，不能包含私有信息（必需）

在发送可见性为公共的通知时，私有用户信息不得包含在通知中。 当通知的可见性为公开时，即使屏幕被锁定，通知中的信息也会显示。 这是因为这种通知存在风险，私密信息可能被第三方物理邻近的终端看到和窃取。

VisibilityPrivateNotificationActivity.java

```
// Prepare a Notification for public display (to be displayed on
 locked screens) that does not contain sensitive information
Notification.Builder publicNotificationBuilder = new Notificatio
n.Builder(this).setContentTitle("Notification : Public");
publicNotificationBuilder.setVisibility(Notification.VISIBILITY_
PUBLIC);
// Do not include private information in Notifications for publi
c display (to be displayed on locked screens)
publicNotificationBuilder.setContentText("Visibility Public: sen
ding notification without sensitive information ");
publicNotificationBuilder.setSmallIcon(R.drawable.ic_launcher);
```

4.10.2.3 对于包含私有信息的通知，可见性必须显式设置为私有或秘密（必需）

即使屏幕锁定，使用 Android 5.0（API Level 21）或更高版本的终端也会显示通知。 因此，当通知包含私有信息时，其可见性标志应显式设置为私有或秘密。 这是为了防止通知中包含的私有信息显示在锁定屏幕上。

目前，可见性的默认值被设置为私有，所以前述风险只有在该标志显式变为公共时才会出现。 但是，可见性的默认值可能会在未来发生变化; 出于这个原因，并且为了在处理信息时始终清楚地表达意图，必须对包含私有信息的通知，将可见性显式设置为私有。

VisibilityPrivateNotificationActivity.java

```
// Create a Notification that includes private information
Notification.Builder priavteNotificationBuilder = new Notificati
on.Builder(this).setContentTitle("Notification : Private");
// *** POINT *** Explicitly set Visibility=Private when creating
 the Notification
priavteNotificationBuilder.setVisibility(Notification.VISIBILITY
_PRIVATE);
```

私有信息的典型示例包括发送给用户的电子邮件，用户的位置数据，以及“5.5 处理隐私数据”部分列出的其他项目。

在使用 Android 4.3（API 级别 18）或更高版本的终端上，用户可以使用“设置”窗口，授予应用读取通知的权限，授予此权限的应用将能够读取通知中的所有信息；因此，除私有用户信息以外的敏感信息不得包含在通知中。

4.10.2.4 使用可见性为私有的通知，创建可见性为公共的额外通知用于展示（推荐）

当传递可见性为私有的信息时，最好同时创建一个额外的通知，用于公开展示，它的可见性为公开；这是为了限制锁定屏幕上显示的信息。

如果公开显示的通知未与可见性为私有的通知一起注册，则在屏幕锁定时将显示由操作系统准备的默认消息。 因此在这种情况下没有安全问题。 但是，为了在处理信息时始终清晰地表达意图，建议显示创建并注册公开显示的通知。

VisibilityPrivateNotificationActivity.java

```
// Create a Notification that contains private information
Notification.Builder privateNotificationBuilder = new Notificati
on.Builder(this).setContentTitle("Notification : Private");
// *** POINT *** Explicitly set Visibility=Private when creating
 the Notification
if (Build.VERSION.SDK_INT >= 21)
    privateNotificationBuilder.setVisibility(Notification.VISIBI
LITY_PUBLIC);
// *** POINT *** Notifications with Visibility=Private may inclu
de private information
privateNotificationBuilder.setContentText("Visibility Private :
Including user info.");
privateNotificationBuilder.setSmallIcon(R.drawable.ic_launcher);
// When creating a Notification with Visibility=Private, simulta
neously create and register a public-display Notification with V
isibility=Public
if (Build.VERSION.SDK_INT >= 21)
    privateNotificationBuilder.setPublicVersion(publicNotificati
on);
```

4.10.3 高级话题

4.10.3 用户授予的查看通知的权限

如上面“4.10.2.1 无论可见性设置如何，通知不得包含敏感信息（尽管私人信息是例外）”所述，在使用 Android 4.3（API Level 18）或更高版本的终端上，某些用户选择的应用，已被授予用户权限，可能会读取所有通知中的信息。

但是，为了使应用有资格获得此用户权限，应用必须实现从 `NotificationListenerService` 派生的服务。

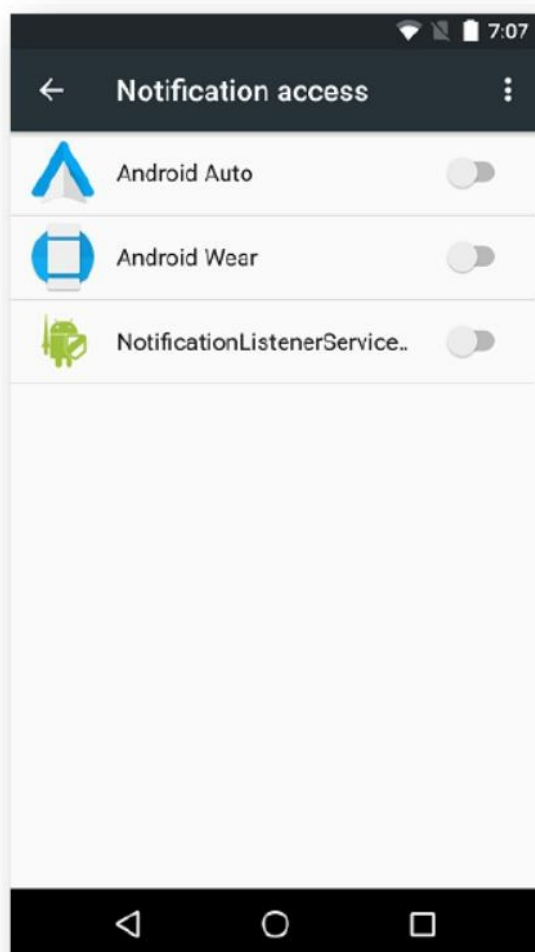


Figure 4.10-3 The Access to Notifications window, from which Notification read controls may be configured

下面的代码展示了 `NotificationListenerService` 的用法。

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.notification.notificationListenerService">

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <service android:name=".MyNotificationListenerService"
        android:label="@string/app_name"
        android:permission="android.permission.BIND_NOTIFICATION
_LISTENER_SERVICE">
            <intent-filter>
                <action android:name=
                "android.service.notification.NotificationListen
erService" />
            </intent-filter>
        </service>
    </application>
</manifest>
```

MyNotificationListenerService.java

```
package org.jssec.notification.notificationListenerService;

import android.app.Notification;
import android.service.notification.NotificationListenerService;
import android.service.notification.StatusBarNotification;
import android.util.Log;

public class MyNotificationListenerService extends NotificationL
istenerService {

    @Override
    public void onNotificationPosted(StatusBarNotification sbn)
{
        // Notification is posted.
        outputNotificationData(sbn, "Notification Posted : ");
    }

    @Override
    public void onNotificationRemoved(StatusBarNotification sbn)
{
        // Notification is deleted.
        outputNotificationData(sbn, "Notification Deleted : ");
    }

    private void outputNotificationData(StatusBarNotification sb
n, String prefix) {
        Notification notification = sbn.getNotification();
        int notificationID = sbn.getId();
        String packageName = sbn.getPackageName();
        long PostTime = sbn.getPostTime();
        String message = prefix + "Visibility :" + notification.
visibility + " ID : " + notificationID;
        message += " Package : " + packageName + " PostTime : "
+ PostTime;
        Log.d("NotificationListen", message);
    }
}
```

如上所述，通过使用 `NotificationListenerService` 获取用户权限，可以读取通知。 但是，由于通知中终端上包含的信息经常包含私有信息，因此在处理此类信息时需要小心。

五、如何使用安全功能

Android 中准备了各种安全功能，如加密，数字签名和权限等。如果这些安全功能使用不当，安全功能无法有效工作，并且会存在漏洞。 本章将解释如何正确使用安全功能。

5.1 创建密码输入界面

5.1.1 示例代码

创建密码输入界面时，这里描述了安全性方面需要考虑的一些要点。 这里仅提及与密码输入有关的内容。 对于如何保存密码，未来会发布另一篇文章。

要点：

1) 输入的密码应该被屏蔽显示（用 `*` 显示）

2) 提供以纯文本显示密码的选项。

3) 警告用户以纯文本显示密码有风险。

要点：处理最后输入的密码时，请注意以下几点以及上述要点。

4) 如果在初始界面中有最后输入的密码，则将黑点的固定数字显示为虚拟，以便不会猜到最后的密码的数字。

5) 当显示虚拟密码，并按下“显示密码”按钮时，清除最后输入的密码并提供输入新密码的状态。

6) 当最后输入的密码显示为虚拟时，如果用户尝试输入密码，请清除最后输入的密码，并将新的用户输入视为新密码。

password_activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:padding="10dp" >
    <!-- Label for password item -->
    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/password" />
    <!-- Label for password item -->
    <!-- *** POINT 1 *** The input password must be masked (Disp
lay with black dot) -->
    <EditText
    android:id="@+id/password_edit"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/hint_password"
    android:inputType="textPassword" />
    <!-- *** POINT 2 *** Provide the option to display the passw
```

```
ord in a plain text -->
    <CheckBox
    android:id="@+id/password_display_check"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/display_password" />
    <!-- *** POINT 3 *** Alert a user that displaying password i
n a plain text has a risk. -->
    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/alert_password" />
    <!-- Cancel/OK button -->
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:gravity="center"
        android:orientation="horizontal" >
        <Button
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:onClick="onClickCancelButton"
        android:text="@android:string/cancel" />
        <Button
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:onClick="onClickOkButton"
        android:text="@android:string/ok" />
    </LinearLayout>
</LinearLayout>
```

位于 `PasswordActivity.java` 底部的 3 个方法的实现，应该取决于目的而调整。

- `private String getPreviousPassword()`
- `private void onClickCancelButton(View view)`
- `private void onClickOkButton(View view)`

PasswordActivity.java

```
package org.jssec.android.password.passwordinputui;

import android.app.Activity;
import android.os.Bundle;
import android.text.Editable;
import android.text.InputType;
import android.text.TextWatcher;
import android.view.View;
import android.view.WindowManager;
```

```
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.EditText;
import android.widget.Toast;

public class PasswordActivity extends Activity {

    // Key to save the state
    private static final String KEY_DUMMY_PASSWORD = "KEY_DUMMY_
PASSWORD";
    // View inside Activity
    private EditText mPasswordEdit;
    private CheckBox mPasswordDisplayCheck;

    // Flag to show whether password is dummy display or not
    private boolean mIsDummyPassword;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.password_activity);
        // Set Disabling Screen Capture
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_SEC
URE);
        // Get View
        mPasswordEdit = (EditText) findViewById(R.id.password_ed
it);
        mPasswordDisplayCheck = (CheckBox) findViewById(R.id.pas
sword_display_check);
        // Whether last Input password exist or not.
        if (getPreviousPassword() != null) {
            // *** POINT 4 *** In the case there is the last inp
ut password in an initial display,
            // display the fixed digit numbers of black dot as d
ummy in order not that the digits number of last password is gue
ssed.
            // Display should be dummy password.
            mPasswordEdit.setText("**********");
            // To clear the dummy password when inputting passwo
rd, set text change listener.
            mPasswordEdit.addTextChangedListener(new PasswordEdi
tTextWatcher());
            // Set dummy password flag
            mIsDummyPassword = true;
        }
        // Set a listner to change check state of password displ
ay option.
        mPasswordDisplayCheck
            .setOnCheckedChangeListener(new OnPasswordDisplayChe
ckedChangeListener());
    }

    @Override
```

```
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        // Unnecessary when specifying not to regenerate Activit
y by the change in screen aspect ratio.
        // Save Activity state
        outState.putBoolean(KEY_DUMMY_PASSWORD, mIsDummyPassword
);
    }

    @Override
    public void onRestoreInstanceState(Bundle savedInstanceState)
 {
        super.onRestoreInstanceState(savedInstanceState);
        // Unnecessary when specifying not to regenerate Activit
y by the change in screen aspect ratio.
        // Restore Activity state
        mIsDummyPassword = savedInstanceState.getBoolean(KEY_DUM
MY_PASSWORD);
    }

    /**
    * Process in case password is input
    */
    private class PasswordEditTextWatcher implements TextWatcher
{

        public void beforeTextChanged(CharSequence s, int start,
int count, int after) {
            // Not used
        }

        public void onTextChanged(CharSequence s, int start, int
 before, int count) {
            // *** POINT 6 *** When last Input password is displ
ayed as dummy, in the case an user tries to input password,
            // Clear the last Input password, and treat new user
 input as new password.
            if (mIsDummyPassword) {
                // Set dummy password flag
                mIsDummyPassword = false;
                // Trim space
                CharSequence work = s.subSequence(start, start +
 count);
                mPasswordEdit.setText(work);
                // Cursor position goes back the beginning, so b
ring it at the end.
                mPasswordEdit.setSelection(work.length());
            }
        }

        public void afterTextChanged(Editable s) {
            // Not used
        }
```

```
    }

    /**
    * Process when check of password display option is changed.
    */
    private class OnPasswordDisplayCheckedChangeListener impleme
nts OnCheckedChangeListener {

        public void onCheckedChanged(CompoundButton buttonView,
boolean isChecked) {
            // *** POINT 5 *** When the dummy password is displa
yed and the "Show password" button is pressed,
            // clear the last input password and provide the sta
te for new password input.
            if (mIsDummyPassword && isChecked) {
                // Set dummy password flag
                mIsDummyPassword = false;
                // Set password empty
                mPasswordEdit.setText(null);
            }
            // Cursor position goes back the beginning, so memor
ize the current cursor position.
            int pos = mPasswordEdit.getSelectionStart();
            // *** POINT 2 *** Provide the option to display the
 password in a plain text
            // Create InputType
            int type = InputType.TYPE_CLASS_TEXT;
            if (isChecked) {
                // Plain display when check is ON.
                type |= InputType.TYPE_TEXT_VARIATION_VISIBLE_PA
SSWORD;
            } else {
                // Masked display when check is OFF.
                type |= InputType.TYPE_TEXT_VARIATION_PASSWORD;
            }
            // Set InputType to password EditText
            mPasswordEdit.setInputType(type);
            // Set cursor position
            mPasswordEdit.setSelection(pos);
        }
    }

    // Implement the following method depends on application
    /**
    * Get the last Input password
    *
    * @return Last Input password
    */
    private String getPreviousPassword() {
        // When need to restore the saved password, return passw
ord character string
        // For the case password is not saved, return null
        return "hirake5ma";
```

```
    }

    /**
    * Process when cancel button is clicked
    *
    * @param view
    */
    public void onClickCancelButton(View view) {
        // Close Activity
        finish();
    }

    /**
    * Process when OK button is clicked
    *
    * @param view
    */
    public void onClickOkButton(View view) {
        // Execute necessary processes like saving password or u
sing for authentication
        String password = null;
        if (mIsDummyPassword) {
            // When dummy password is displayed till the final m
oment, grant last iInput password as fixed password.
            password = getPreviousPassword();
        } else {
            // In case of not dummy password display, grant the
user input password as fixed password.
            password = mPasswordEdit.getText().toString();
        }
        // Display password by Toast
        Toast.makeText(this, "password is ¥"" + password + "¥"",
 Toast.LENGTH_SHORT).show();
        // Close Activity
        finish();
    }
}
```

5.1.2 规则书

实现密码输入界面时，遵循以下规则。

5.1.2.1 如果输入了密码，提供屏蔽显示功能（必需）

智能手机通常用在火车或公共汽车等拥挤的地方，而且存在密码被某人偷窥的风险。 因此，屏蔽显示密码的功能是应用规范所必需的。

有两种方法可以将 `EditText` 显示为密码：在布局 XML 中静态指定此值，或通过从程序中切换显示来动态指定此值。 前者通过为 `android:inputType` 属性指定 `textPassword` 或使用 `android:password` 属性来实现。 后者通过使用 `EditText` 类的 `setInputType()` 方法，将 `InputType.TYPE_TEXT_VARIATION_PASSWORD` 添加到其输入类型，来实现的。

下面展示了每个的示例代码。

在布局 XML 中屏蔽密码。

password_activity.xml

```
<!—Password input item -->
<!—Set true for the android:password attribute -->
<EditText
    android:id="@+id/password_edit"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/hint_password"
    android:password="true" />
```

在活动中屏蔽密码。

PasswordActivity.java

```
// Set password display type
// Set TYPE_TEXT_VARIATION_PASSWORD for InputType.
EditText passwordEdit = (EditText) findViewById(R.id.password_ed
it);
int type = InputType.TYPE_CLASS_TEXT
    | InputType.TYPE_TEXT_VARIATION_PASSWORD;
passwordEdit.setInputType(type);
```

5.1.2.2 提供以纯文本展示密码的选项（必需）

智能手机的密码输入通过触摸面板输入完成，因此与 PC 上的键盘输入相比，容易发生误输入。由于输入不便，用户可能会使用简单的密码，这样做会更危险。此外，当有多次密码输入失败导致帐户锁定等机制时，必须尽可能避免误输入。作为这些问题的解决方案，通过准备以纯文本显示密码的选项，用户可以使用安全密码。

但是，以纯文本显示密码时，可能会被嗅探，所以使用此选项时。有必要提醒用户注意来自后面的嗅探。此外，如果存在以纯文本显示的选项，则还需要为系统准备，来自动取消纯文本显示，如设置纯文本显示的时间。密码纯文本显示的限制，在未来版本的另一篇文章中发布。因此，密码纯文本显示的限制不包含在示例代码中。

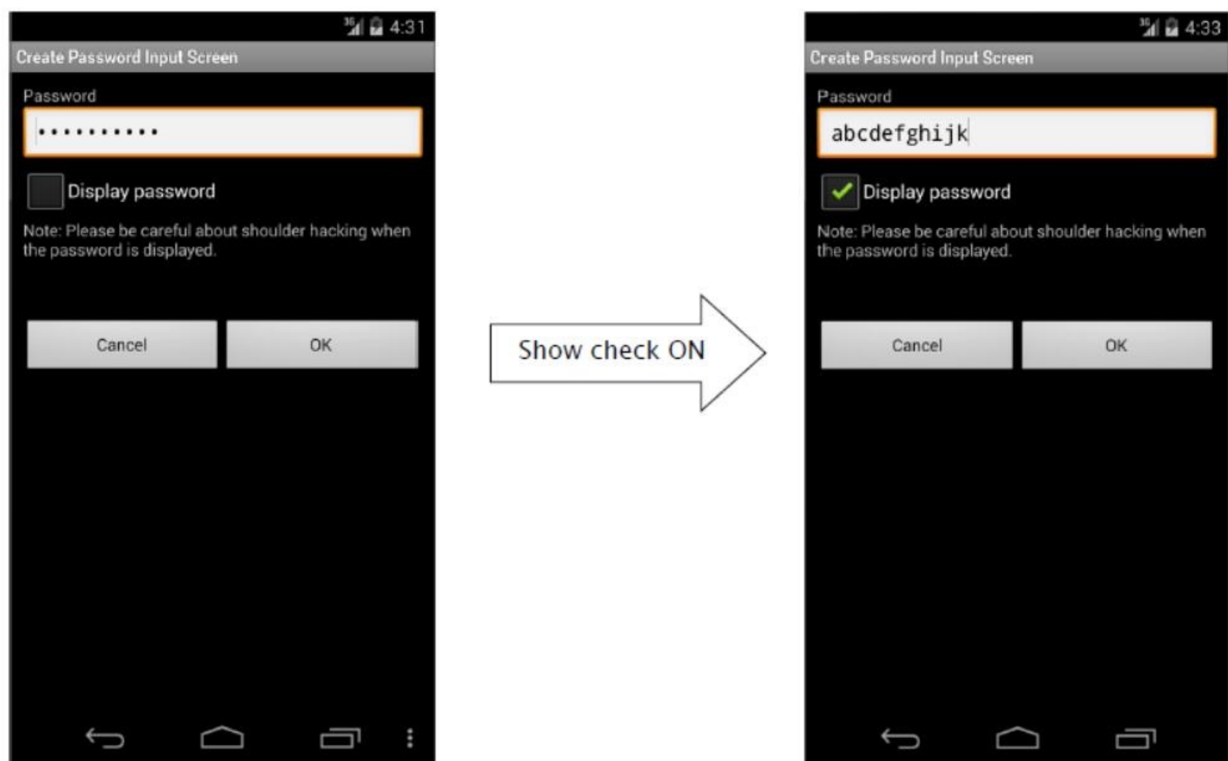


Figure 5.1-2

通过指定 `EditText` 的 `InputType` ，可以切换屏蔽显示和纯文本显示。

PasswordActivity.java

```
/**
* Process when check of password display option is changed.
*/
private class OnPasswordDisplayCheckedChangeListener implements
    OnCheckedChangeListener {

    public void onCheckedChanged(CompoundButton buttonView,
    boolean isChecked) {
        // *** POINT 5 *** When the dummy password is displayed
and the "Show password" button is pr
        essed,
        // Clear the last input password and provide the state f
or new password input.
        if (mIsDummyPassword && isChecked) {
            // Set dummy password flag
            mIsDummyPassword = false;
            // Set password empty
            mPasswordEdit.setText(null);
        }
        // Cursor position goes back the beginning, so memorize
the current cursor position.
        int pos = mPasswordEdit.getSelectionStart();
        // *** POINT 2 *** Provide the option to display the pas
sword in a plain text
        // Create InputType
        int type = InputType.TYPE_CLASS_TEXT;
        if (isChecked) {
            // Plain display when check is ON.
            type |= InputType.TYPE_TEXT_VARIATION_VISIBLE_PASSWO
RD;
        } else {
            // Masked display when check is OFF.
            type |= InputType.TYPE_TEXT_VARIATION_PASSWORD;
        }
        // Set InputType to password EditText
        mPasswordEdit.setInputType(type);
        // Set cursor position
        mPasswordEdit.setSelection(pos);
    }
}
```

5.1.2.3 活动加载时屏蔽密码（必需）

为防止密码被偷窥，当活动启动时，密码显示选项的默认值应该设置为 `OFF` 。基本上，默认值应该总是定义为更安全的一方。

5.1.2.4 显示最后输入密码时，必须显示虚拟密码（必需）

当指定最后输入的密码时，不要给第三方任何密码提示，它应该显示为带有屏蔽字符（ * 等）的固定位数的虚拟值。 另外，在虚拟显示时按下“显示密码”的情况下，清除密码并切换到纯文本显示模式。 它有助于防止最后输入的密码被嗅探的风险，即使设备被传递给第三方，比如它被盗时。 仅供参考，在虚拟显示的情况下以及用户尝试输入密码时，应取消虚拟显示，需要变成正常输入状态。

显示最后输入的密码时，显示虚拟密码。

PasswordActivity.java

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.password_activity);
    // Get View
    mPasswordEdit = (EditText) findViewById(R.id.password_edit);
    mPasswordDisplayCheck = (CheckBox) findViewById(R.id.passwor
d_display_check);
    // Whether last Input password exist or not.
    if (getPreviousPassword() != null) {
        // *** POINT 4 *** In the case there is the last input p
assword in an initial display,
        // display the fixed digit numbers of black dot as dummy
 in order not that the digits number of last password is guessed.

        // Display should be dummy password.
        mPasswordEdit.setText("**********");
        // To clear the dummy password when inputting password,
set text change listener.
        mPasswordEdit.addTextChangedListener(new PasswordEditTex
tWatcher());
        // Set dummy password flag
        mIsDummyPassword = true;
    }

    [...]

    }

/**
* Get the last input password.
*
* @return the last input password
*/
private String getPreviousPassword() {
    // To restore the saved password, return the password charac
ter string.
    // For the case password is not saved, return null.
    return "hirake5ma";
}
```

在虚拟显示的情况下，当密码显示选项打开时，请清除显示的内容。

PasswordActivity.java

```
/**
* Process when check of password display option is changed.
*/
private class OnPasswordDisplayCheckedChangeListener implements
    OnCheckedChangeListener {

    public void onCheckedChanged(CompoundButton buttonView,
        boolean isChecked) {
        // *** POINT 5 *** When the dummy password is displayed
and the "Show password" button is pressed,
        // Clear the last input password and provide the state f
or new password input.
        if (mIsDummyPassword && isChecked) {
            // Set dummy password flag
            mIsDummyPassword = false;
            // Set password empty
            mPasswordEdit.setText(null);
        }

        [...]

    }
}
```

在虚拟显示的情况下，当用户尝试输入密码时，清除虚拟显示。

PasswordActivity.java

```
// Key to save the state
private static final String KEY_DUMMY_PASSWORD = "KEY_DUMMY_PASS
WORD";

[...]

// Flag to show whether password is dummy display or not.
private boolean mIsDummyPassword;

@Override
public void onCreate(Bundle savedInstanceState) {
    [...]

    // Whether last Input password exist or not.
    if (getPreviousPassword() != null) {
        // *** POINT 4 *** In the case there is the last input p
assword in an initial display,
        // display the fixed digit numbers of black dot as dummy
 in order not that the digits number of last password is guessed.
```

```
        // Display should be dummy password.
        mPasswordEdit.setText("**********");
        // To clear the dummy password when inputting password,
set text change listener.
        mPasswordEdit.addTextChangedListener(new PasswordEditTex
tWatcher());
        // Set dummy password flag
        mIsDummyPassword = true;
    }

    [...]

}

@Override
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    // Unnecessary when specifying not to regenerate Activity by
 the change in screen aspect ratio.
    // Save Activity state
    outState.putBoolean(KEY_DUMMY_PASSWORD, mIsDummyPassword);
}

@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    // Unnecessary when specifying not to regenerate Activity by
 the change in screen aspect ratio.
    // Restore Activity state
    mIsDummyPassword = savedInstanceState.getBoolean(KEY_DUMMY_P
ASSWORD);
}

/**
* Process when inputting password.
*/
private class PasswordEditTextWatcher implements TextWatcher {

    public void beforeTextChanged(CharSequence s, int start, int
 count,
        int after) {
        // Not used
    }

    public void onTextChanged(CharSequence s, int start, int bef
ore,
        int count) {
        // *** POINT 6 *** When last Input password is displayed
 as dummy, in the case an user tries to input password,
        // Clear the last Input password, and treat new user inp
ut as new password.
        if (mIsDummyPassword) {
```

```
            // Set dummy password flag
            mIsDummyPassword = false;
            // Trim space
            CharSequence work = s.subSequence(start, start + cou
nt);
            mPasswordEdit.setText(work);
            // Cursor position goes back the beginning, so bring
 it at the end.
            mPasswordEdit.setSelection(work.length());
        }
    }

    public void afterTextChanged(Editable s) {
        // Not used
    }
}
```

5.1.3 高级话题

5.1.3.1 登录过程

需要密码输入的代表性示例是登录过程。以下是一些在登录过程中需要注意的事项。

登录失败时的错误信息

在登录过程中，需要输入两个信息，ID（账号）和密码。 登录失败时有两种情况。 一个是 ID 不存在。 另一个是 ID 存在，但密码不正确。 如果这两种情况中的任何一种，有所区分并显示在登录失败消息中，则攻击者可以猜测指定的 ID 是否存在。 为了阻止这种猜测，这两种情况不应该在登录失败消息中区分，并且该消息应该按照下面的方式显示。

消息示例：登录 ID 或密码不正确。

自动登录功能

存在一个功能，可以完成成功登录过程一次后，通过省略下次登录的 ID /密码输入来执行自动登录。自动登录功能可以省去复杂的输入。因此，便利性会增加，但另一方面，当智能手机被盗时，第三方恶意使用的风险将随之而来。

只有在恶意第三方造成的损害可以接受时，或者只有在可以采取足够安全措施的情况下，才能使用自动登录功能。例如，在网上银行应用的情况下，当设备由第三方运营时，可能会造成财务损失。所以在这种情况下，与自动登录功能配套的安全措施是必需的。存在一些可能的应对措施，例如【在付款过程等财务流程前需要重新输入密码】，【设置自动登录时，请求用户注意并提示用户锁定设备】等。使用自动登录时，有必要仔细考虑方便性和风险以及假定的对策。

5.1.3.2 修改密码

更改曾经设置的密码时，应在屏幕上准备以下输入项目。

- 当前密码
- 新密码
- 新密码（确认）

当引入自动登录功能时，第三方可能使用应用。 在这种情况下，为了避免意外更改密码，需要输入当前的密码。 另外，为了减少由于错误输入新密码，而进入不可用状态的风险，有必要要求输入两次新的密码。

5.1.3.3 关于“使密码可见”设置

Android 设置菜单中有一个名为“使密码可见”的设置。 在 Android 4.4 的情况下，如下所示。

```
设置 -> 安全 -> 使密码可见
```

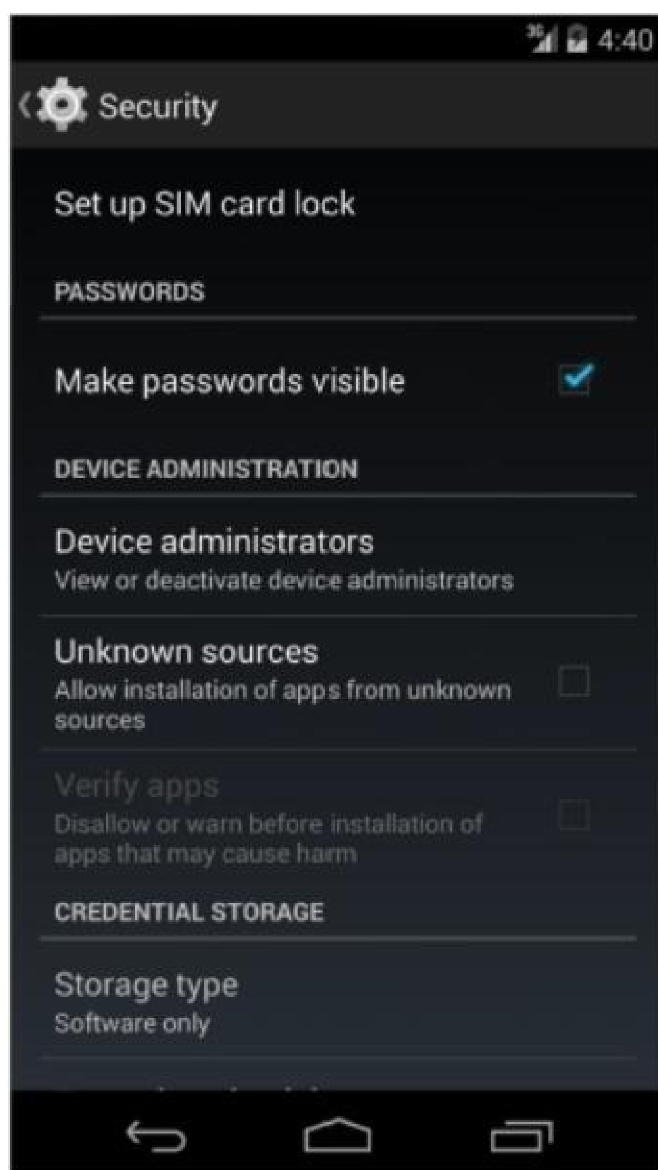


Figure 5.1-3

打开“使密码可见”设置时，最后输入的字符以纯文本显示。 经过一定的时间（约两秒），或输入下一个字符后，以纯文本显示的字符将被屏蔽。 关闭时，输入后会立即屏蔽。 此设置影响整个系统，并且它适用于使用 `EditText` 的密码显示功能的所有应用。

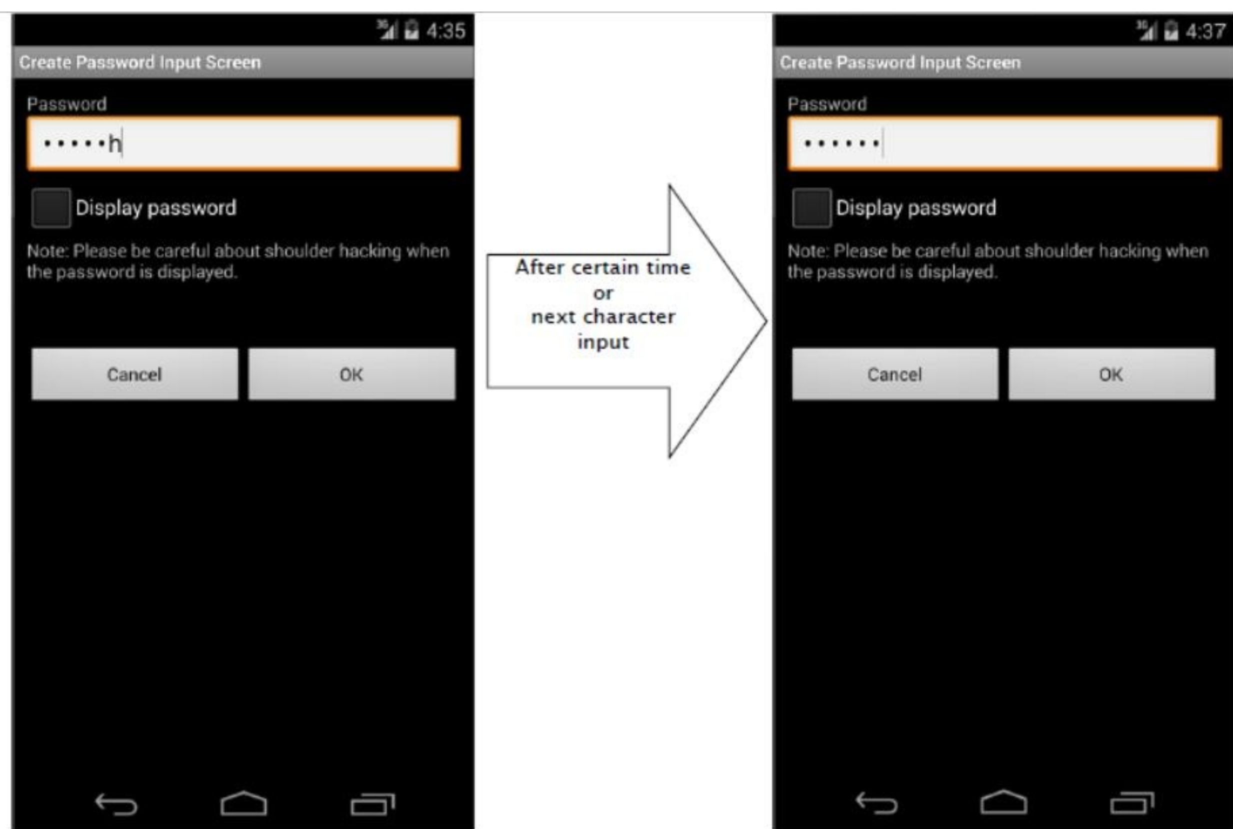


Figure 5.1-4

5.1.3.4 禁用屏幕截图

在密码输入屏幕中，密码可以在屏幕上清晰显示。 在处理个人信息的屏幕中，如果屏幕截图功能在默认情况下处于启用状态，则可能会从屏幕截图文件中泄漏，它存储在外部存储器上。因此建议对密码输入屏幕禁用屏幕截图功能。 通过附加下面的代码可以禁用屏幕截图。

PasswordActivity.java

```
@Override
public void onCreate(Bundle saveInstanceState) {
    [...]

    Window window = getWindow();
    window.addFlags(WindowManager.LayoutParams.FLAG_SECURE);
    setContentView(R.layout.passwordInputScreen);

    [...]
}
```

5.2 权限和保护级别

权限内有四种类型的保护级别，它们包括正常，危险，签名和签名或系统。 根据保护级别，权限被称为正常权限，危险权限，签名权限或签名或系统权限。 以下部分中使用这些名称。

5.2.1 示例代码

5.2.1.1 如何使用 Android OS 的系统权限

Android 操作系统有一个称为“权限”的安全机制，可以保护其用户的资产（如联系人和 GPS 功能）免受恶意软件的侵害。 当应用请求访问受 Android OS 保护的信息或功能时，应用需要显式声明权限才能访问它们。 安装应用，它申请需要用户同意的权限时，会出现以下确认界面 [23]。

> [23] 在 Android 6.0（API Level 23）及更高版本中，安装应用时不会发生用户的权限授予或拒绝，而是在应用请求权限时在运行时发生。 更多详细信息，请参见“5.2.1.4 在 Android 6.0 及更高版本中使用危险权限的方法”和“5.2.3.6 Android 6.0 和更高版本中的权限模型规范的修改”部分。

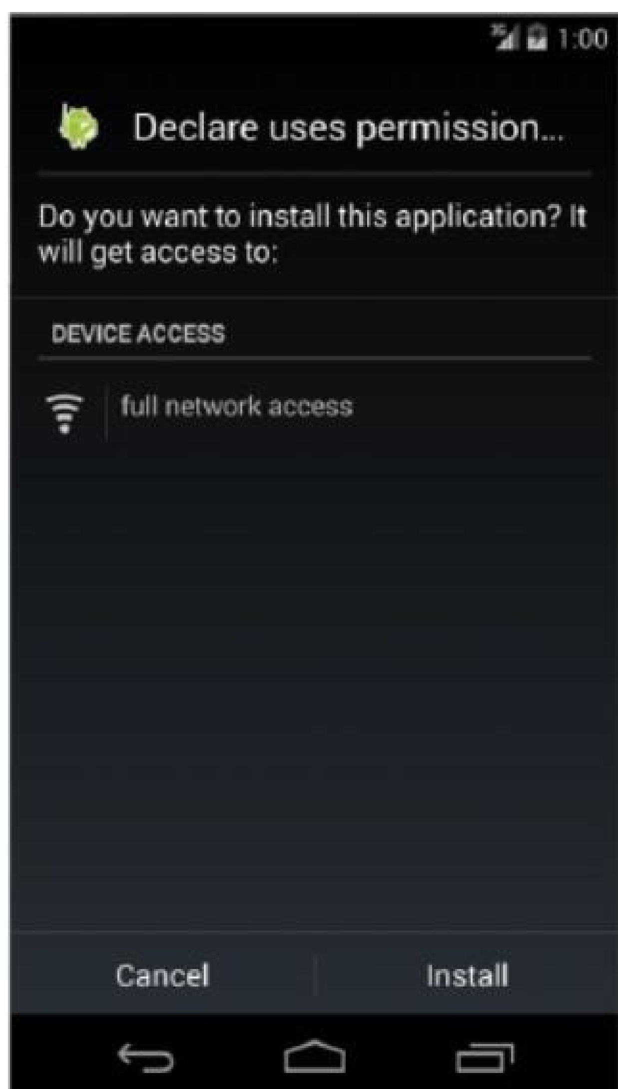


Figure 5.2-1

从该确认界面中，用户能够知道，应用试图访问哪些类型的特征或信息。 如果应用试图访问明显不需要的功能或信息，那么该应用很可能是恶意软件。 因此，为了使你的应用不被怀疑是恶意软件，因此需要尽量减少使用权限声明。

要点：

1. 使用 `uses-permission` 声明应用中使用的权限。
2. 不要用 `uses-permission` 声明任何不必要的权限。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.permission.usespermission" >
    <!-- *** POINT 1 *** Declare a permission used in an applica
tion with uses-permission -->
    <!-- Permission to access Internet -->
    <uses-permission android:name="android.permission.INTERNET"/>

    <!-- *** POINT 2 *** Do not declare any unnecessary permissi
ons with uses-permission -->
    <!-- If declaring to use Permission that is unnecessary for
application behaviors, it gives users a
    sense of distrust. -->
    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

5.2.1.2 如何使用内部定义的权限在内部应用之间通信

除了由 Android OS 定义的系统权限之外，应用还可以定义自己的权限。 如果使用内部定义的权限（内部定义的签名权限更准确），则可以构建只允许内部应用之间进行通信的机制。 通过提供基于多个内部应用之间的，应用间通信的复合功能，应用变得更具吸引力，你的企业可以通过将其作为系列销售获得更多利润。 这是使用内部定义的签名权限的情况。

示例应用“内部定义的签名权限（UserApp）”使用 `Context.startActivity()` 方法启动示例应用“内部定义的签名权限（ProtectedApp）”。 两个应用都需要使用相同的开发人员密钥进行签名。如果用于签名的密钥不同，则 `UserApp` 不会向 `ProtectedApp` 发送意图，并且 `ProtectedApp` 不处理从 `UserApp` 收到的意图。 此外，它还可以防止恶意软件使用安装顺序相关的事项，绕过你自己的签名权限，如高级话题部分中所述。

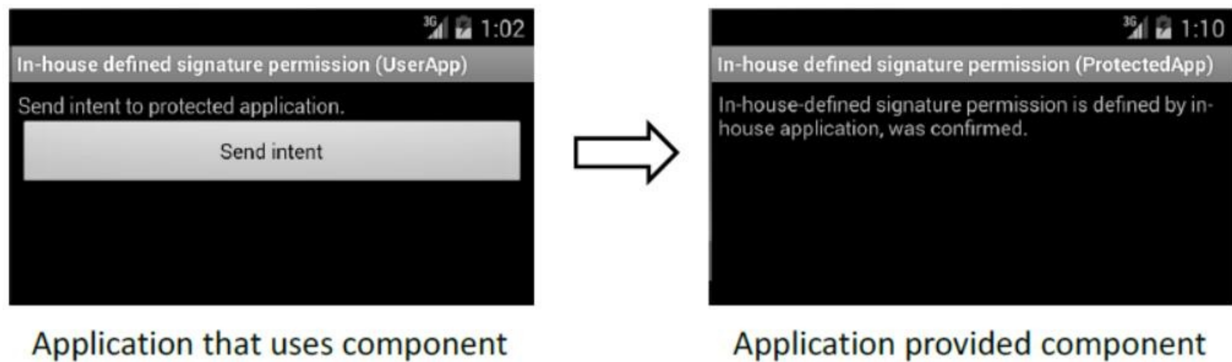


Figure 5.2-2

要点：提供组件的应用

1) 使用 `protectionLevel="signature"` 定义权限。

2) 对于组件，使用其权限属性强制规定权限。

3) 如果组件是活动，则必须没有定义意图过滤器。

4) 在运行时，验证签名权限是否由程序代码本身定义。

5) 导出 APK 时，请使用与使用该组件的应用相同的开发人员密钥对 APK 进行签名。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.permission.protectedapp" >
    <!-- *** POINT 1 *** Define a permission with protectionLeve
l="signature" -->
    <permission
        android:name="org.jssec.android.permission.protectedapp.
MY_PERMISSION"
        android:protectionLevel="signature" />
    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <!-- *** POINT 2 *** For a component, enforce the permis
sion with its permission attribute -->
        <activity
            android:name=".ProtectedActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:permission="org.jssec.android.permission.pro
tectedapp.MY_PERMISSION" >
            <!-- *** POINT 3 *** If the component is an activity
, you must define no intent-filter -->
        </activity>
    </application>
</manifest>
```

ProtectedActivity.java

```
package org.jssec.android.permission.protectedapp;

import org.jssec.android.shared.SigPerm;
import org.jssec.android.shared.Utils;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.widget.TextView;

public class ProtectedActivity extends Activity {

    // In-house Signature Permission
    private static final String MY_PERMISSION = "org.jssec.andro
id.permission.protectedapp.MY_PERMISSION";
    // Hash value of in-house certificate
    private static String sMyCertHash = null;

    private static String myCertHash(Context context) {
        if (sMyCertHash == null) {
            if (Utils.isDebuggable(context)) {
                // Certificate hash value of "androiddebugkey" o
```

```
f debug.keystore
                sMyCertHash = "0EFB7236 328348A9 89718BAD DF57F5
44 D5CCB4AE B9DB34BC 1E29DD26 F77C8255";
            } else {
                // Certificate hash value of "my company key" of
 keystore
                sMyCertHash = "D397D343 A5CBC10F 4EDDEB7C A10062
DE 5690984F 1FB9E88B D7B3A7C2 42E142CA";
            }
        }
        return sMyCertHash;
    }

    private TextView mMessageView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mMessageView = (TextView) findViewById(R.id.messageView)
;
        // *** POINT 4 *** At run time, verify if the signature
permission is defined by itself on the program code
        if (!SigPerm.test(this, MY_PERMISSION, myCertHash(this))
) {
            mMessageView.setText("In-house defined signature per
mission is not defined by in-house application.");
            return;
        }
        // *** POINT 4 *** Continue processing only when the cer
tificate matches
        mMessageView.setText("In-house-defined signature permiss
ion is defined by in-house application,
        was confirmed.");
    }
}
```

SigPerm.java

```
package org.jssec.android.shared;

import android.content.Context;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.PermissionInfo;

public class SigPerm {

    public static boolean test(Context ctx, String sigPermName,
String correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, sigPermName));
    }

    public static String hash(Context ctx, String sigPermName) {
        if (sigPermName == null) return null;
        try {
            // Get the package name of the application which dec
lares a permission named sigPermName.
            PackageManager pm = ctx.getPackageManager();
            PermissionInfo pi;
            pi = pm.getPermissionInfo(sigPermName, PackageManage
r.GET_META_DATA);
            String pkgname = pi.packageName;
            // Fail if the permission named sigPermName is not a
 Signature Permission
            if (pi.protectionLevel != PermissionInfo.PROTECTION_
SIGNATURE) return null;
            // Return the certificate hash value of the applicat
ion which declares a permission named sigPermName.
            return PkgCert.hash(ctx, pkgname);
        } catch (NameNotFoundException e) {
            return null;
        }
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;
```

```
public class PkgCert {

    public static boolean test(Context ctx, String pkgname, Stri
ng correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
    }
}
```

要点 5：导出 APK 时，请使用与使用该组件的应用相同的开发人员密钥对 APK 进行签名。

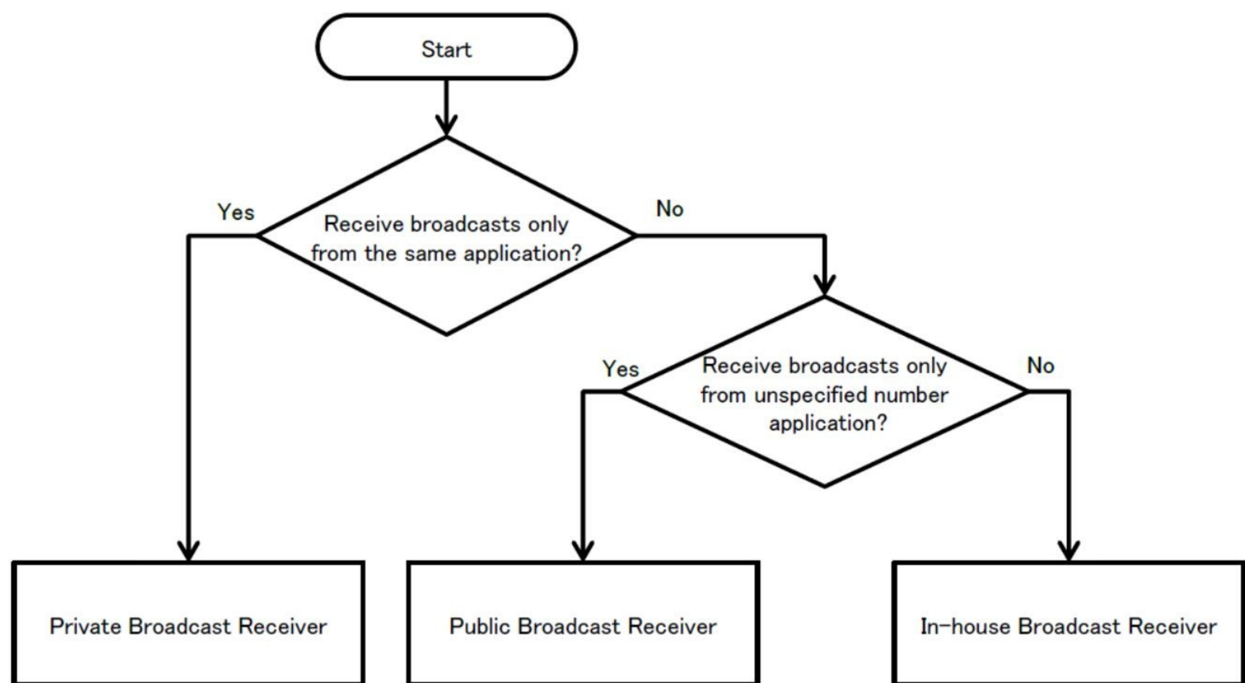


Figure 4.2-1

要点：使用组件的应用

6) 禁止定义应用使用的相同签名权限。

7) 使用权限标签声明内部权限。

8) 验证内部签名权限，是否由提供组件的应用定义。

9) 验证目标应用是否是内部应用。

10) 当目标组件是一个活动时，使用显式意图。

11) 导出 APK 时，请使用与使用该组件的应用相同的开发人员密钥对 APK 进行签名。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.permission.userapp" >
    <!-- *** POINT 6 *** The same signature permission that the
application uses must not be defined -->
    <!-- *** POINT 7 *** Declare the in-house permission with us
es-permission tag -->
    <uses-permission
    android:name="org.jssec.android.permission.protectedapp.MY_P
ERMISSION" />
    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
        android:name=".UserActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

UserActivity.java

```
package org.jssec.android.permission.userapp;

import org.jssec.android.shared.PkgCert;
import org.jssec.android.shared.SigPerm;
import org.jssec.android.shared.Utils;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class UserActivity extends Activity {

    // Requested (Destination) application's Activity information

    private static final String TARGET_PACKAGE = "org.jssec.andr
oid.permission.protectedapp";
    private static final String TARGET_ACTIVITY = "org.jssec.and
```

```
roid.permission.protectedapp.ProtectedActivity";
    // In-house Signature Permission
    private static final String MY_PERMISSION = "org.jssec.andro
id.permission.protectedapp.MY_PERMISSION";
    // Hash value of in-house certificate
    private static String sMyCertHash = null;

    private static String myCertHash(Context context) {
        if (sMyCertHash == null) {
            if (Utils.isDebuggable(context)) {
                // Certificate hash value of "androiddebugkey" o
f debug.keystore.
                sMyCertHash = "0EFB7236 328348A9 89718BAD DF57F5
44 D5CCB4AE B9DB34BC 1E29DD26 F77C8255";
            } else {
                // Certificate hash value of "my company key" of
 keystore.
                sMyCertHash = "D397D343 A5CBC10F 4EDDEB7C A10062
DE 5690984F 1FB9E88B D7B3A7C2 42E142CA";
            }
        }
        return sMyCertHash;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void onSendButtonClicked(View view) {
        // *** POINT 8 *** Verify if the in-house signature perm
ission is defined by the application that provides the component
 on the program code.
        if (!SigPerm.test(this, MY_PERMISSION, myCertHash(this))
) {
            Toast.makeText(this, "In-house-defined signature per
mission is not defined by In house application.", Toast.LENGTH_L
ONG).show();
            return;
        }
        // *** POINT 9 *** Verify if the destination application
 is an in-house application.
        if (!PkgCert.test(this, TARGET_PACKAGE, myCertHash(this)
)) {
            Toast.makeText(this, "Requested (Destination) applic
ation is not in-house application.", Toast.LENGTH_LONG).show();
            return;
        }
        // *** POINT 10 *** Use an explicit intent when the dest
ination component is an activity.
        try {
            Intent intent = new Intent();
```

```
            intent.setClassName(TARGET_PACKAGE, TARGET_ACTIVITY)
;
            startActivity(intent);
        } catch(Exception e) {
            Toast.makeText(this,
            String.format("Exception occurs:%s", e.getMessage())
, Toast.LENGTH_LONG).show();
        }
    }
}
```

PkgCertWhitelists.java

```
package org.jssec.android.shared;

import java.util.HashMap;
import java.util.Map;
import android.content.Context;

public class PkgCertWhitelists {

    private Map<String, String> mWhitelists = new HashMap<String
, String>();

    public boolean add(String pkgname, String sha256) {
        if (pkgname == null) return false;
        if (sha256 == null) return false;
        sha256 = sha256.replaceAll(" ", "");
        if (sha256.length() != 64) return false; // SHA-256 -> 3
2 bytes -> 64 chars
        sha256 = sha256.toUpperCase();
        if (sha256.replaceAll("[0-9A-F]+", "").length() != 0) re
turn false; // found non hex char
        mWhitelists.put(pkgname, sha256);
        return true;
    }

    public boolean test(Context ctx, String pkgname) {
        // Get the correct hash value which corresponds to pkgna
me.
        String correctHash = mWhitelists.get(pkgname);
        // Compare the actual hash value of pkgname with the cor
rect hash value.
        return PkgCert.test(ctx, pkgname, correctHash);
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;

public class PkgCert {

    public static boolean test(Context ctx, String pkgname, Stri
ng correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
```

```
    }
}
```

要点 11：导出 APK 时，请使用与使用该组件的应用相同的开发人员密钥对 APK 进行签名。

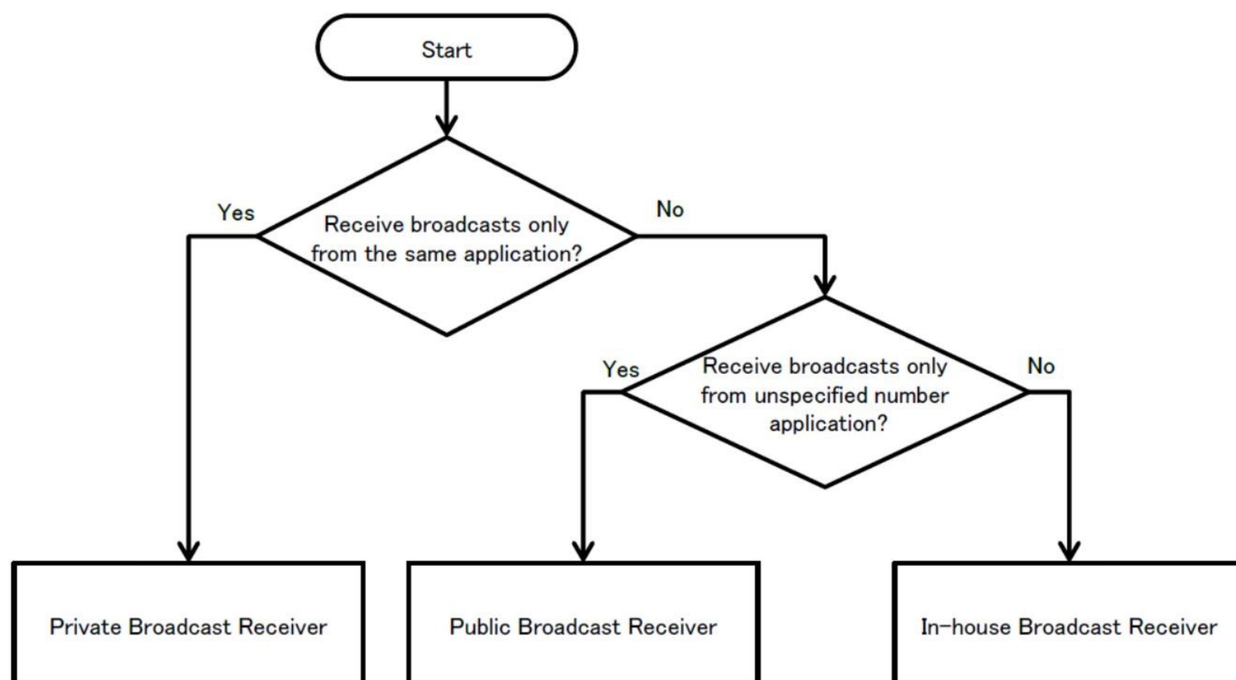


Figure 4.2-1

5.2.1.3 如何验证应用证书的散列值

我们将说明，如何验证应用证书的散列值，他们在本指南中不同位置出现。 严格来说，散列值意味着“用于签署 APK 的开发人员密钥的公钥证书的 SHA256 散列值”。

如何使用 Keytool 进行验证

使用与 JDK 捆绑在一起的名为 keytool 的程序，你可以获取开发人员密钥的公钥证书的散列值（也称为证书指纹）。 由于散列算法的不同，存在各种散列方法，例如 MD5，SHA1 和 SHA256。 但是，考虑到加密字节长度的安全强度，本指南推荐使用 SHA256。 不幸的是， Android SDK 中使用的， JDK6 绑定的 keytool 不支持 SHA256 来计算哈希值。 因此，有必要使用 JDK7 绑定的 keytool。

通过 keytool 输出 Android 调试证书内容的示例

```
> keytool -list -v -keystore < keystore file > -storepass < pass
word >
Type of keystore: JKS
Keystore provider: SUN
One entry is included in a keystore
Other name: androiddebugkey
Date of creation: 2012/01/11
Entry type: PrivateKeyEntry
Length of certificate chain: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 4f0cef98
Start date of validity period: Wed Jan 11 11:10:32 JST 2012 End
date: Fri Jan 03 11:10:32 JST 2042
Certificate fingerprint:
MD5: 9E:89:53:18:06:B2:E3:AC:B4:24:CD:6A:56:BF:1E:A1
SHA1: A8:1E:5D:E5:68:24:FD:F6:F1:ED:2F:C3:6E:0F:09:A3:07:F8:5C:0
C
SHA256: FB:75:E9:B9:2E:9E:6B:4D:AB:3F:94:B2:EC:A1:F0:33:09:74:D8
:7A:CF:42:58:22:A2:56:85:1B:0F
:85:C6:35
Signatrue algorithm name: SHA1withRSA
Version: 3
*******************************************
*******************************************
```

如何使用 JSSEC 证书散列值检查器进行验证

在不安装JDK7的情况下，你可以使用 JSSEC 证书散列值检查器，轻松验证证书散列值。

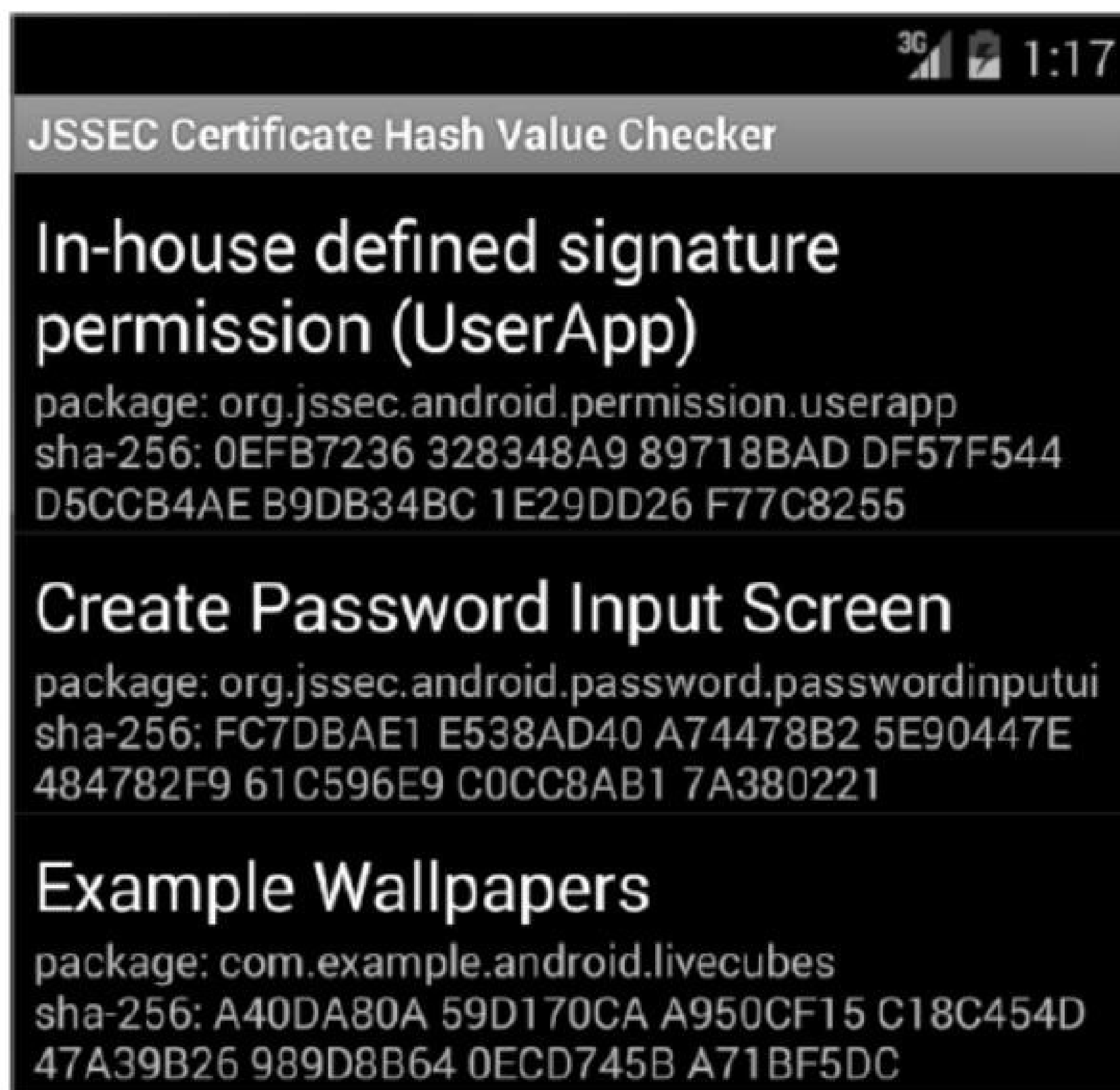


Figure 5.2-5

这是一个 Android 应用，显示安装在设备中的，应用的证书哈希值列表。 在上图中， `sha-256` 右侧显示的 64 个字符的十六进制字符串是证书哈希值。 本指南附带的示例代码文件夹 `JSSEC CertHash Checker` 是一组源代码。 如果你愿意，你可以编译代码并使用它。

5.2.1.4 Android 6.0 及更高版本中使用危险权限的方法

Android 6.0（API Level 23）结合了修改后的规范，与应用实现相关 - 特别是应用被授予权限的时间。

在 Android 5.1（API 级别 22）和更早版本的权限模型下（请参阅“5.2.3.6 Android 6.0 和更高版本中的权限模型规范修改”一节），安装时授予应用申请的所有权限 。 但是，在 Android 6.0 及更高版本中，应用开发人员必须以这样的方式实现应用，即对于危险权限，应用在适当的时候请求权限。 当应用请求权限时，Android OS 会向用户显示如下所示的确认窗口，请求用户决定，是否授予相关权限。 如果用户允许使用权限，则应用可以执行任何需要该权限的操作。



该规范还修改了权限授予的单位。 以前，所有权限都是同时授予的；在 Android 6.0（API Level 23）及更高版本中，权限是单独授予的（按权限组）。 结合这种修改，用户现在可以看到每个权限的单独确认窗口，允许用户在授予权限或拒绝权限时，作出更灵活的决定。 应用开发人员必须重新审视其应用的规格和设计，并充分考虑到权限被拒绝的可能性。

Android 6.0 及更高版本中的权限模型的详细信息，请参见“5.2.3.6 Android 6.0 和更高版本中的权限模型规范修改”部分。

要点：

1) 应用声明他们将使用的权限

2) 不要声明不必要的权限

3) 检查是否应用被授予了权限

4) 请求权限（打开一个对话框来向用户请求权限）

5) 对拒绝使用权限的情况实现适当的行为

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.permission.permissionrequestingpe
rmissionatruntime" >
    <!-- *** POINT 1 *** Apps declare the Permissions they will 
use -->
    <!-- Permission to read information on contacts (Protection 
Level: dangerous) -->
    <uses-permission android:name="android.permission.READ_CONTA
CTS" />
    <!-- *** POINT 2 *** Do not declare the use of unnecessary P
ermissions -->
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
            <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".ContactListActivity"
            android:exported="false">
        </activity>
    </application>
</manifest>
```

MainActivity.java

```
package org.jssec.android.permission.permissionrequestingpermiss
ionatruntime;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
```

```
public class MainActivity extends AppCompatActivity implements V
iew.OnClickListener {

    private static final int REQUEST_CODE_READ_CONTACTS = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        readContacts();
    }

    private void readContacts() {
        // *** POINT 3 *** Check whether or not Permissions have
 been granted to the app
        if (ContextCompat.checkSelfPermission(getApplicationCont
ext(), Manifest.permission.READ_CONTACTS) != PackageManager.PERM
ISSION_GRANTED) {
            // Permission was not granted
            // *** POINT 4 *** Request Permissions (open a dialo
g to request permission from users)
            ActivityCompat.requestPermissions(this, new String[]
{Manifest.permission.READ_CONTACTS}, REQUEST_CODE_READ_CONTACTS)
;
        } else {
            // Permission was previously granted
            showContactList();
        }
    }

    // A callback method that receives the result of the user's
selection
    @Override
    public void onRequestPermissionsResult(int requestCode, Stri
ng[] permissions, int[] grantResults) {
        switch (requestCode) {
            case REQUEST_CODE_READ_CONTACTS:
                if (grantResults.length > 0 && grantResults[0] =
= PackageManager.PERMISSION_GRANTED) {
                    // Permissions were granted; we may execute
operations that use contact information
                    showContactList();
                } else {
                    // Because the Permission was denied, we may
 not execute operations that use contact information
                    // *** POINT 5 *** Implement appropriate beh
avior for cases in which the use of a Permission is refused
```

```
                    Toast.makeText(this, String.format("Use of c
ontact is not allowed."), Toast.LENGTH_LONG).show();
                }
            return;
        }
    }

    // Show contact list
    private void showContactList() {
        // Launch ContactListActivity
        Intent intent = new Intent();
        intent.setClass(getApplicationContext(), ContactListActi
vity.class);
        startActivity(intent);
    }
}
```

5.2.2 规则书

使用内部权限时，请确保遵循以下规则：

5.2.2.1 Android 的系统危险权限只能用于保护用户资产（必需）

由于不建议你使用自己的危险权限（请参阅“5.2.2.2 你自己的危险权限不得使用（必需）”），我们将在使用 Android 操作系统的系统危险权限的前提下进行。

不像其他三种类型的权限，危险权限具有这个特性，需要用户同意授予应用权限，在声明了危险权限的设备上安装应用时，将显示以下屏幕：随后， 用户可以知道应用试图使用的权限级别（危险权限和正常权限），当用户点击“安装”时，应用将被授予权限，然后安装。

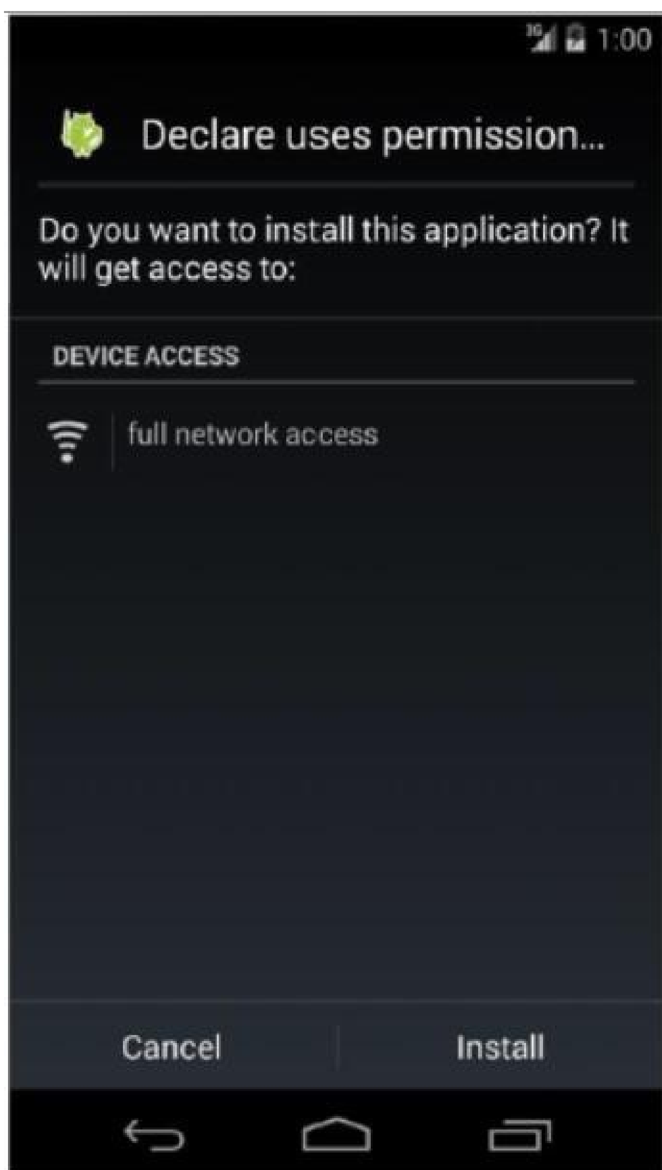


Figure 5.2-7

应用可以处理开发人员希望保护的用户资产。 我们必须意识到，危险的权限只能保护用户资产，因为用户只是授予权限的人。 另一方面，开发人员想要保护的资产不能用上述方法保护。

例如，假设应用具有一个组件，只与内部应用通信，它不允许从其他公司的任何应用访问该组件，并且通过危险权限的保护来实现。 当用户根据判断，向另一家公司的应用授予权限时，需要保护的内部资产可能通过应用授权来利用。 为了在此类情况下保护内部资产，我们建议使用内部定义的签名权限。

5.2.2.2 不能使用你自己的危险权限（必需）

即使使用内部定义的危险权限，在某些情况下，屏幕提示“请求允许来自用户的权限”也不会显示。 这意味着，有时根据用户判断来请求权限的特性（危险权限的特征）不起作用。 因此，指导手册规定“不得使用内部定义的危险权限”。

为了解释它，我们假设有两种类型的应用。 第一种类型的应用定义了内部危险权限，并且它让受此权限保护的组件公开。 我们称之为 `ProtectedApp` 。 另一个是我们称为 `AttackerApp` ，它试图利用 `ProtectedApp` 的组件。 我们还假设 `AttackerApp` 不仅声明了使用它的权限，而且还定义了相同的权限。

在以下情况下， `AttackerApp` 可以在未经用户同意的情况下，使用 `ProtectedApp` 的组件：

1. 当用户安装 `AttackerApp` 时，安装将在没有屏幕提示的情况下完成，它要求用户授予应用危险权限。
2. 同样，当用户安装 `ProtectedApp` 时，安装将会完成而没有任何特别的警告。
3. 当用户启动 `AttackerApp` 后， `AttackerApp` 可以访问 `ProtectedApp` 的组件，而不会被用户检测到，这可能会导致损失。

这种情况的原因在下面解释。 当用户尝试首先安装 `AttackerApp` 时，在特定设备上，尚未使用 `uses-permission` 来定义声明的权限。 没有发现错误，Android 操作系统将继续安装。 由于只有在安装时用户才需要同意危险权限，因此已安装的应用将被视为已被授予权限。 因此，如果稍后安装的应用的组件受到名称相同的危险权限的保护，则在未经用户同意的情况下，事先安装的应用将能够利用该组件。

此外，由于在安装应用时，确保存在 Android OS 定义的系统危险权限，每次安装具有 `uses-permission` 的应用时，都会显示用户验证提示。 只有在自定义危险权限的情况下才会出现此问题。 在写这篇文章的时候，还没有开发出可行方法，在这种情况下保护组件的访问。 因此，你不得使用你自己的危险权限。

5.2.2.3 你自己的签名权限必需仅在提供方定义（必需）

如“5.2.1.2 如何使用内部定义的签名权限，在内部应用之间进行通信”中所示，在进行内部应用之间的内部通信时，通过检查签名权限，可以确保安全性。 当使用这种机制时，保护级别为签名的权限的定义，必须写在具有组件的提供方应用的 `AndroidManifest.xml` 中，但用户方应用不能定义签名权限。

此规则也适用于 `signatureOrSystem` 权限。原因如下。

我们假设，在提供方应用之前安装了多个用户方应用，并且每个用户方应用，不仅要求提供方应用定义的签名权限，而且还定义了相同的权限。 在这些情况下，所有用户方应用都可以在安装提供方应用之后，立即访问提供方应用。 随后，卸载先安装的用户方应用时，权限的定义也将被删除，然后该权限将变为未定义。 因此，其余的用户方应用将无法访问提供方应用。

以这种方式，当用户方应用定义了一个自定义权限时，它可能会意外地将权限设置为未定义。因此，只有提供需要保护的组件的提供方应用才应该定义权限，并且必须避免在用户方定义权限。

通过如上所述的那样，自定义权限将在安装提供方应用时由 Android OS 应用，并且在卸载应用时权限将变得未定义。因此，由于权限定义总是对应提供方应用的定义，因此可以提供适当的组件并对其进行保护。请注意，这个观点成立，是因为对于内部定义的签名权限，用户方应用被授予权限，而不管应用在相互通信中的安装顺序 [24]。

> [24] 如果使用正常/危险权限，并且用户方应用安装在提供方应用之前，则该权限将不会授予用户方应用，权限仍未定义。 因此，即使在安装了提供方应用之后，也不能访问组件。

5.2.2.4 验证内部定义的签名权限是否由内部应用定义（必需）

实际上，只有通过 `AnroidManifest.xml` 声明签名权限并使用权限来保护组件，才能说是足够安全。 此问题的详细信息，请参阅“高级主题”部分中的“5.2.3.1 绕过自定义签名权限的 Android 操作系统特性及其对策”。

以下是安全并正确使用内部定义的签名权限的步骤。

首先，在 `AndroidManifest.xml` 中编写如下代码：

在提供方应用的 `AndroidManifest.xml` 中定义内部签名权限。（权限定义）

例如： <permission android:name="xxx" android:protectionLevel="signatur

在提供方应用的 `AndroidManifest.xml` 中，使用要保护的组件的权限属性强制执行权限。 （执行权限）

例如： <activity android:permission="xxx" ... >...</activity>

在每个用户方应用的 `AndroidManifest.xml` 中，使用 `uses-permission` 标签声明内部定义的签名权限，来访问要保护的组件。 （使用权限声明）

例如： <uses-permission android:name="xxx" />

下面，在源代码中实现这些：

在处理组件的请求之前，首先验证内部定义的签名权限是否由内部应用定义。 如果不是，请忽略该请求。 （保护提供方组件）

在访问组件之前，请先验证内部定义的签名权限是否由内部应用定义。 否则，请勿访问组件（用户方组件中的保护）。

最后，使用 Android Studio 的签名功能之前，执行下列事情：

使用相同的开发人员密钥，对所有互相通信的应用的 APK 进行签名。

在此，对于如何实现“确认内部定义签名权限已由内部应用定义”的具体要点，请参阅“5.2.1.2 如何使用内部定义的签名权限，在内部应用之间进行通信”。

此规则也适用于 `signatureOrSystem` 权限。

5.2.2.5 不应该使用你自己的普通权限（推荐）

应用只需在 `AndroidManifest.xml` 中使用 `uses-permission` 声明，即可使用正常权限。 因此，你不能使用正常权限，来保护组件免受恶意软件的安装。

此外，在使用自定义普通权限进行应用间通信的情况下，应用是否可以被授予权限取决于安装顺序。 例如，当你安装已声明使用普通权限的应用（用户方法），并且在另一应用（提供者端）之前，它拥有已定义权限的组件，用户方应用将无法 访问受权限保护的组件，即使稍后安装提供方应用也是如此。

作为一种方法，防止由于安装顺序而导致的应用间通信丢失，你可以考虑在通信中的每个应用中定义权限。 通过这种方式，即使在提供方应用之前安装了用户方应用，所有用户方应用也将能够访问提供方应用。 但是，它会产生一种情况，即在卸载第一个安装的用户方应用时，权限未定义。 因此，即使有其他用户方应用，他们也无法访问提供方应用。

如上所述，存在损害应用可用性的风险，因此不应使用你自己的正常权限。

5.2.2.6 你自己的权限名称的字符串应该是应用包名的扩展（推荐）

当多个应用使用相同名称定义权限时，将使用先安装的应用所定义的保护级别。 如果首先安装的应用定义了正常权限，并且稍后安装的应用使用相同的名称定义了签名权限，则签名权限的保护将不可用。 即使没有恶意的意图，多个应用之间的权限名称冲突，也可能导致任何应用的行为成为意外的保护级别。 为防止发生此类事故，建议权限名称扩展于定义权限的应用的包名（以它开头），如下所示。

```
(package name).permission.(identifying string)
```

例如，为 `org.jssec.android.sample` 包定义 `READ` 访问权限时，以下名称将是首选。

```
org.jssec.android.sample.permission.READ
```

5.2.3 高级话题

5.2.3.1 绕过自定义签名许可的 Android 操作系统特性及其对策

自定义签名权限是一种权限，实现使用相同开发人员密钥签名的应用之间的应用间通信。 由于开发人员密钥是私钥，不能公开，因此只有在内部应用互相通信的情况下，才有权使用签名权限进行保护。

首先，我们将描述在 Android 的开发者指南（http://developer.android.com/guide/topics/security/security.html）中解释的自定义签名权限的基本用法。 但是，后面将解释，存在绕过许可方面的问题。 因此，本指南中描述的对策是必要的。

以下是自定义签名权限的基本用法。

在提供方应用的 `AndroidManifest.xml` 中定义内部签名权限。（权限定义）

例如： `<permission android:name="xxx" android:protectionLevel="signatur`

在提供方应用的 `AndroidManifest.xml` 中，使用要保护的组件的权限属性强制执行权限。 （执行权限）

例如： `<activity android:permission="xxx" ... >...</activity>`

在每个用户方应用的 `AndroidManifest.xml` 中，使用 `uses-permission` 标签声明内部定义的签名权限，来访问要保护的组件。 （使用权限声明）

例如： `<uses-permission android:name="xxx" />`

使用相同的开发人员密钥，对所有互相通信的应用的 APK 进行签名。

实际上，如果满足以下条件，这种方法会存在漏洞，可以绕过签名权限。

为了便于说明，我们将受自定义签名权限保护的应用称为 `ProtectedApp` ，并且 `AttackerApp` 是已由不同于 `ProtectedApp` 的开发人员密钥签名的应用。 绕过签名权限的漏洞意味着，即使 `AttackerApp` 的签名不匹配，也有可能访问 `ProtectedApp` 的组件。

条件 1：

`AttackerApp` 也定义了正常权限，与 `ProtectedApp` 所定义的签名权限名称相同（严格来说，签名权限也是可以接受的）。

例如： `<permission android:name=" xxx" android:protectionLevel="normal'`

条件 2：

`AttackerApp` 使用 `uses-permission` 声明了自定义的正常权限。

例如： `<uses-permission android:name="xxx" />`

条件 3：

`AttackerApp` 安装在 `ProtectedApp` 之前。

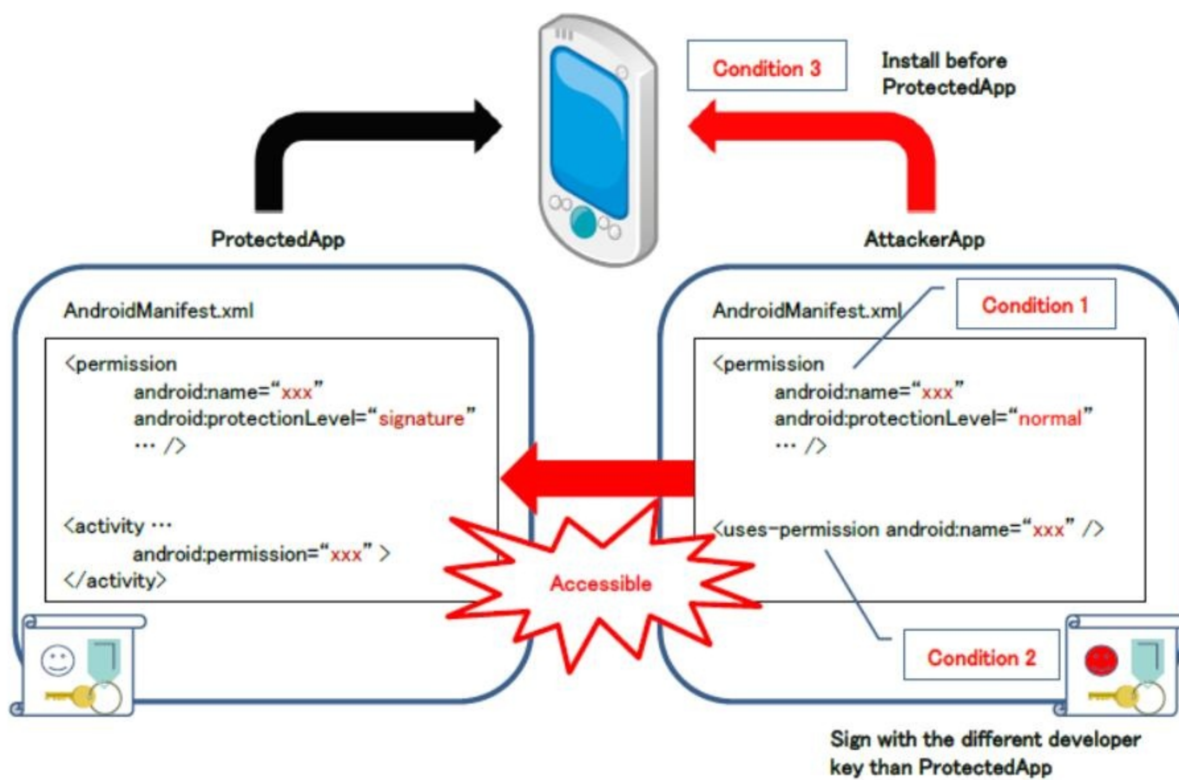


Figure 5.2-8

满足条件 1 和条件 2 所需的权限名称，很容易从 APK `AndroidManifest.xml` 文件中取出，被攻击者知道。 攻击者也可以用一定的努力满足条件 3（例如欺骗用户）。

如果只采用基本用法，就有自定义签名权限的绕过风险，需要采取防范此类漏洞的对策。 具体而言，你可以通过使用“5.2.2.4 验证内部定义的签名权限是否由内部应用定义”中描述的方法来发现如何解决上述问题。

5.2.3.2 用户伪造的 `AndroidManifest.xml`

我们已经谈到，自定义权限的保护级别可能会被改变。 为了防止由于这种情况导致的故障，需要在 Java 的源代码一侧实施某些对策。 从 `AndroidManifest.xml` 伪造的角度来看，我们将讨论在源代码方面要采取的对策。 我们将演示一个可以检测伪造的简单安装案例。 但请注意，对于出于犯罪目的而伪造的专业黑客来说，这些对策效果甚微。

这部分内容关于应用伪造和恶意用户。 尽管这本来不属于指导手册的范围，但由于这与权限有关，并且这种伪造的工具作为 Android 应用公开提供，所以我们决定将其称为“针对业余黑客的简单对策”。

必须记住的是，可以从市场安装的应用，是可以在没有 root 权限的情况下，被伪造的应用。原因是应用可以重建和签署 `AndroidManifest.xml` 文件。通过使用这些应用，任何人都可以删除已安装应用的任何权限。

举个例子，似乎有些情况下重建的 APK 具有不同的签名， `AndroidManifest.xml` 发生改变，并删除了 `INTERNET` 权限，来使应用中附加的广告模块失效。有些用户称赞这些类型的工具，因为任何个人信息没有被泄漏到任何地方。由于这些附加在应用中的广告停止运作，此类行为会对依靠广告收入的开发者造成金钱损失。而且相信大多数用户没有任何反感。

在下面的代码中，我们展示了一个实现的实例，一个使用 `uses-permission` 声明了 `INTERNET` 权限的应用，验证 `INTERNET` 权限是否在运行时在 `AndroidManifest.xml` 文件中描述。

```
public class CheckPermissionActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        // Acquire Permission defined in AndroidManifest.xml
        List<String> list = getDefinedPermissionList();
        // Detect falsification
        if( checkPermissions(list) ){
            // OK
            Log.d("dbg", "OK.");
        }else{
            Log.d("dbg", "manifest file is stale.");
            finish();
        }
    }

    /**
    * Acquire Permission through list that was defined in Androi
dManifest.xml
    * @return
    */
    private List<String> getDefinedPermissionList(){
        List<String> list = new ArrayList<String>();
        list.add("android.permission.INTERNET");
        return list;
    }

    /**
    * Verify that Permission has not been changed Permission
    * @param permissionList
    * @return
    */
    private boolean checkPermissions(List<String> permissionList)
{
        try {
```

```
            PackageInfo packageInfo = getPackageManager().getPac
kageInfo(
            getPackageName(), PackageManager.GET_PERMISSIONS);
            String[] permissionArray = packageInfo.requestedPerm
issions;
            if (permissionArray != null) {
                for (String permission : permissionArray) {
                    if(! permissionList.remove(permission)){
                        // Unintended Permission has been added
                        return false;
                    }
                }
            }
            if(permissionList.size() == 0){
                // OK
                return true;
            }
        } catch (NameNotFoundException e) { }
        return false;
    }
}
```

5.2.3.3 APK 伪造的检测

我们在“5.2.3.2 用户伪造的 `AndroidManifest.xml` ”中，解释了用户对权限的伪造检测。但是，应用伪造并不仅限于权限，在许多其他情况下，应用在没有任何源代码更改的情况下被占用。例如，只是通过将资源替换为自己的应用，他们将其他开发人员的应用（伪造）分发到市场中，就好像它们是自己的应用一样。在这里，我们将展示一个更通用的方法，来检测 APK 文件的伪造。

为了伪造 APK，需要将 APK 文件解码为文件夹和文件，修改其内容，然后将其重建为新的 APK 文件。由于伪造者没有原始开发者的密钥，他必须用他自己的钥匙签署新的 APK 文件。由于 APK 的伪造不可避免地会产生签名（证书）的变化，因此可以通过比较 APK 中的证书，和源代码中嵌入的开发人员证书，在运行时检测 APK 是否被伪造。

以下是示例代码。另外，如果使用这个实现示例，专业黑客将能够轻松绕过伪造检测。请注意这是一个简单的实现示例，请将此示例代码应用于你的应用。

要点：

1. 在开始主要操作之前，验证应用的证书是否属于开发人员。

SignatureCheckActivity.java

```
package org.jssec.android.permission.signcheckactivity;

import org.jssec.android.shared.PkgCert;
import org.jssec.android.shared.Utils;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.widget.Toast;

public class SignatureCheckActivity extends Activity {

    // Self signed certificate hash value
    private static String sMyCertHash = null;

    private static String myCertHash(Context context) {
        if (sMyCertHash == null) {
            if (Utils.isDebuggable(context)) {
                // Certificate hash value of "androiddebugkey" o
f debug.
                sMyCertHash = "0EFB7236 328348A9 89718BAD DF57F5
44 D5CCB4AE B9DB34BC 1E29DD26 F77C8255";
            } else {
                // Certificate hash value of "my company key" of
 keystore
                sMyCertHash = "D397D343 A5CBC10F 4EDDEB7C A10062
DE 5690984F 1FB9E88B D7B3A7C2 42E142CA";
            }
        }
        return sMyCertHash;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        // *** POINT 1 *** Verify that an application's certific
ate belongs to the developer before major processing is started
        if (!PkgCert.test(this, this.getPackageName(), myCertHas
h(this))) {
            Toast.makeText(this, "Self-sign match NG", Toast.LEN
GTH_LONG).show();
            finish();
            return;
        }
        Toast.makeText(this, "Self-sign match OK", Toast.LENGTH_
LONG).show();
    }
}
```

PkgCert.java

```java
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;

public class PkgCert {

    public static boolean test(Context ctx, String pkgname, Stri
ng correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
```

```
    }
}
```

5.2.3.4 权限重委托问题

访问联系人或 GPS，它们带有受 Android OS 保护的信息和功能时，应用必须声明使用权限。当所需的权限被授予时，权限被委托给应用，应用将能够访问受权限保护的信息和功能。

根据程序的设计方式，被授予权限的应用可以获取受权限保护的数据。此外，应用可以向另一个应用提供受保护数据，而不必强制确保相同的权限，这无异于，没有权限的应用可以访问受权限保护的数据。这实际上是重新授权，称为权限重新授权问题。因此，只有 Android 的权限机制的规范，才能够管理从来自用程序的，保护数据的直接访问的权限。

图 5.2-9 展示了一个具体的例子。 中心的应用表明，已声明 `android.permission.READ_CONTACTS` 的应用使用它来读取联系人，然后将它们存储到其自己的数据库中。 当已经存储的信息通过内容供应器，提供给另一个应用，而没有任何限制时，就会发生重新授权问题。

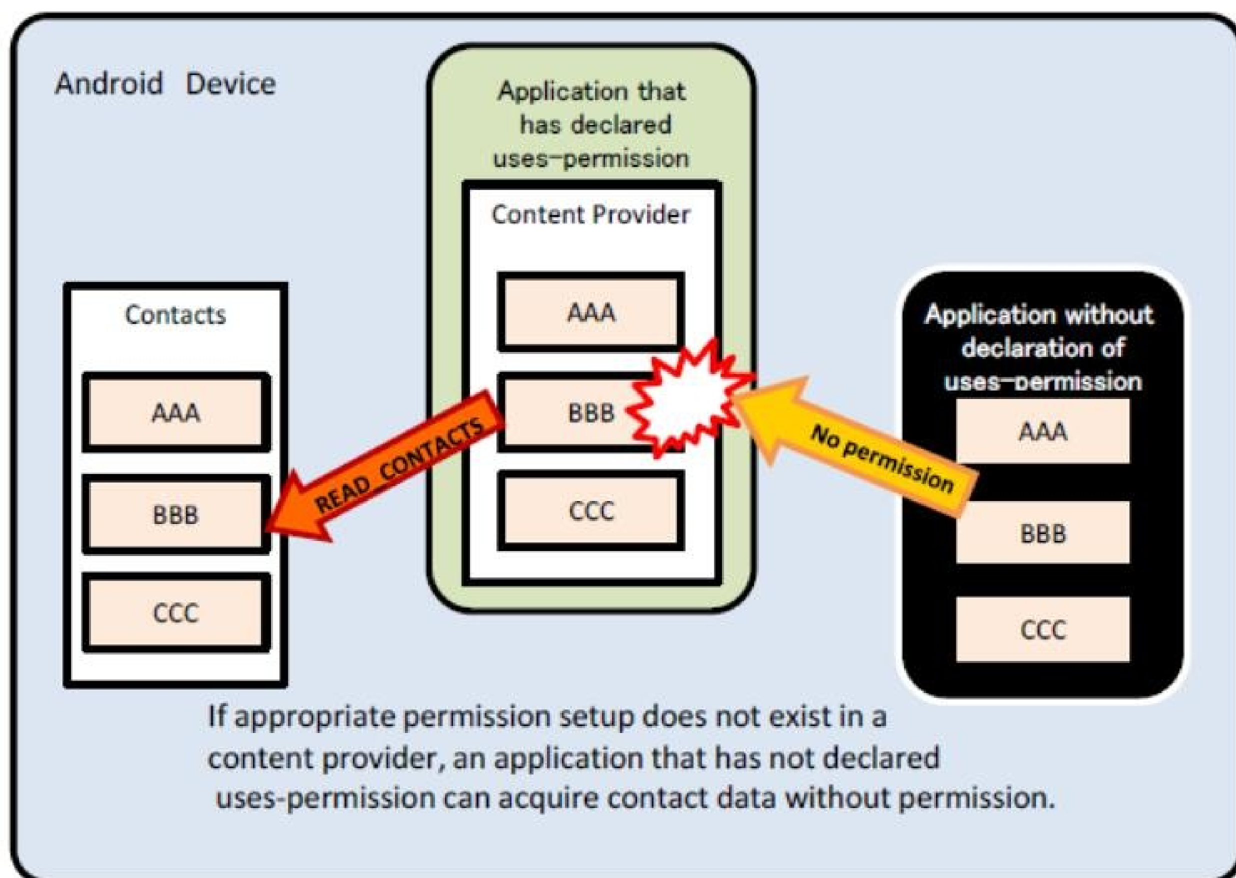


Figure 5.2-9 An Application without Permission Acquires Contacts

作为一个类似的例子，声明了 `android.permission.CALL_PHONE` 的应用，使用它从另一个应用接收电话号码（可能是用户输入的），它未声明相同权限。如果该号码在未经用户验证的情况下被呼叫，那么也存在重新授权问题。

在某些情况下，通过权限获得的，几乎完整的信息或功能资产，需要由其他应用二次提供。在这些情况下，供应方应用必须要求相同权限，才能保持原始的保护级别。此外，在仅以间接方式提供信息和功能资产的一部分的情况下，根据信息或功能资产的一部分的损害程度，需要适当保护。由“4.1.1.1 创建/使用私有活动”或“4.1.1.4 创建/使用私有活动”，我们可以使用类似于前者的保护措施，验证用户的同意，并设置目标应用的活动限制，以及其他。

这种重新授权问题不仅限于 Android 权限。对于 Android 应用，应用从不同的应用，网络和存储介质中获取必要的信息/功能，这是常见的。在很多情况下，访问它们需要一些权限和限制。例如，如果提供者来源的 Android 应用，则它是权限；如果它是网络，那么它是登录机制；如果它是存储介质，则会存在访问限制。因此，在仔细考虑后，需要对应用实现这些措施，因为信息/功能不是以与用户意图相反的方式使用的。以间接方式将获得的信息/功能提供给另一应用，或转移到网络或存储介质时，这一点尤其重要。根据需要，你必须强制确保权限或限制使用权限，如 Android 权限。询问用户的同意是解决方案的一部分。

在以下代码中，我们演示了一个情况，使用 `READ_CONTACTS` 权限，从联系人数据库中获取列表的应用，对信息的目标强制确保相同的 `READ_CONTACTS` 权限。

要点：

强制确保提供者的相同权限。

AndroidManifest.xml

```
<?xml version="1.0" encofing="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.permission.transferpermission"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
    android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.READ_CONTA
CTS"/>
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
        android:name=".TransferPermissionActivity"
        android:label="@string/title_activity_transfer_permissio
n" >
        <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER
" />
        </intent-filter>
        </activity>
        <provider
            android:name=".TransferPermissionContentProvider"
            <!-- *** Point1 *** Enforce the same permission that
 the provider does. -->
            android:authorities="org.jssec.android.permission.tr
ansferpermission"
            android:enabled="true"
            android:exported="true"
            android:readPermission="android.permission.READ_CONT
ACTS" >
        </provider>
    </application>
</manifest>
```

当一个应用确保多个权限时，上述方法不会解决它。 通过使用源代码中的 `Context#checkCallingPermission()` 或 `PackageManager#checkPermission`，它验证调用者应用是否在清单中，使用 `uses-permission` 声明了所有权限。

在活动中：

```
public void onCreate(Bundle savedInstanceState) {
    [...]

    if (checkCallingPermission("android.permission.READ_CONTACTS"
) == PackageManager.PERMISSION_GRANTED
        && checkCallingPermission("android.permission.WRITE_CONT
ACTS") == PackageManager.PERMISSION_GRANTED) {
        // Processing during the time when an invoker is correct
ly declaring to use
        return;
    }
    finish();
}
```

5.2.3.5 自定义权限的签名检查机制（Android 5.0 及以上）

在 Android 5.0（API Level 21）及更高版本中，如果满足以下条件，则无法安装定义其自定义权限的应用。

1. 在设备上已经安装了另一个应用，用相同名称定义了自定义权限。
2. 应用使用不同的密钥签名

当具有受保护函数（组件）的应用，和使用该函数的应用，定义了具有相同名称的自定义权限，并且使用相同密钥签名时，上述机制将防止安装定义了自定义权限的其他公司的应用同名。 但是，如“5.2.2.3 你自己的签名权限必须仅在提供方应用中定义（必需）”中所述，该机制对于检查自定义权限是否由你自己的公司定义是行不通的，因为权限 如果多个应用定义相同的权限，在你自己不知道的情况下，可能通过卸载应用来使其失效。

总而言之，在 Android 5.0（API Level 21）和更高版本中，当你的应用定义你自己的签名权限时，你还需要遵守两个规则：“5.2.2.3 你自己的签名权限只能在提供方应用上定义（必需） “和”5.2.2.4 验证内部定义的签名权限是否由内部应用定义（必需）“。

5.2.3.6 Android 版本 6.0 和更高版本中对权限模型规范的修改

Android 6.0（API Level 23）引入了权限模型的修改规范，这些规范影响了应用的设计和规范。 在本节中，我们将概述 Android 6.0 及更高版本中的权限模型。

权限授予和拒绝的时机

如果应用声明使用需要用户确认的权限（危险权限）【请参见“5.2.2.1 Android 系统危险权限必须仅用于保护用户资产（必需）”一节】，Android 5.1（API 级别 22）和更早的版本，要求在安装应用时显示这些权限的列表，并且用户必须授予所有权限才能继续安装。 此时，应用声明的所有权限（包括危险权限以外的权限）均已授予该应用；一旦这些权限被授予应用，它们就会一直有效，直到应用从终端上卸载。

但是，在 Android 6.0 及更高版本的规范中，应用执行时会授予权限。 在安装应用时不会发生权限授予和用户的权限确认。 当应用执行需要危险权限的过程时，需要检查是否已将这些权限提前授予应用；如果没有，则必须在 Android 操作系统中显示确认窗口，来请求用户的同意 [25]。如果用户从确认窗口授予权限，则将权限授予应用。 但是，用户授予应用的权限（危险权限）可以随时通过设置菜单撤销（图 5.2-10）。 出于这个原因，必须实现适当的过程，来确保应用不会产生不规则的行为，即使在因为未授予权限，而无法访问所需的信息或功能的情况下。

> [25] 由于正常权限和签名权限是由 Android OS 自动授予的，因此不需要获取用户对这些权限的确认。



权限授予和拒绝的单位

根据与之相关的功能和信息类型，可以将多个权限组合在一起称为权限组。 例如，读取日历信息所需的权限 `android.permission.READ_CALENDAR` 以及写入日历信息所需的权限 `android.permission.WRITE_CALENDAR` 都关联权限组 `android.permission-group.CALENDAR` 。

在 Android 6.0 及更高版本的新权限模型中，权限的授予和撤销可以使用权限组统一执行。 因此，当一个应用在运行时请求 `android.permission.READ_CALENDAR` 并且用户同意该请求时，Android OS 的行为就像 `android.permission.READ_CALENDAR` 和 `android.permission.WRITE_CALENDAR` 都已被授权一样。 如果随后请求 `android.permission.WRITE_CALENDAR` 权限，则操作系统不会向用户显示对话框，而是直接授予权限。

权限组分类的更多信息，请参阅开发人员参考（http://developer.android.com/intl/ja/guide/topics/security/permissions.html#perm-groups）。

修改后的规范的影响范围

应用在运行时需要权限请求的情况，仅限于终端运行 Android 6.0 或更高版本，并且应用的 `targetSDKVersion` 为 23 或更高的情况。 如果终端运行的是 Android 5.1 或更低版本，或者应用的 `targetSDKVersion` 为 22 或更低，则安装时会完全请求和授予权限，这与传统情况相同。 但是，如果终端运行的是 Android 6.0 或更高版本，则即使应用的 `targetSDKVersion` 低于 23，用户在安装时授予的权限也可能随时被用户撤销。 这会造成应用意外终止的可能性。 开发人员必须遵守修改后的规范，或将应用的 `maxSDKVersion` 设置为 22 或更低版本，来确保该应用不能安装在运行 Android 6.0（API Level 23）或更高版本（表 5.2-1）的终端上。

表.2-1

Android OS 终端版本	应用的 `targetSDKVersion`	应用被授予权限的时机	用户是否能控制权限
>= 6.0	>= 23	执行时	是
>= 6.0	< 23	安装时	是（需要快速响应）
<= 5.1	>= 23	安装时	否
<= 5.1	< 23	安装时	否

但是，应该注意， `maxSdkVersion` 的影响是有限的。 当 `maxSdkVersion` 的值设置为 22 或更低时，Android 6.0（API Level 23）和更高版本的设备，不再被列为 Google Play 中目标应用的可安装设备。 另一方面，由于未在 Google Play 以外的市场中检查 `maxSdkVersion` 的值，因此可能会在 Android 6.0（API Level 23）或更高版本中安装目标应用。 由于 `maxSdkVersion` 的效果有限，Google 不建议使用 `maxSdkVersion` ，因此建议开发人员立即遵守修改后的规范。

在 Android 6.0 及更高版本中，以下网络通信权限的保护级别从危险更改为正常。 因此，即使应用声明使用这些权限，也不需要获得用户的显式统一，因此修改后的规范在此情况下不会产生影响。

- `android.permission.BLUETOOTH`
- `android.permission.BLUETOOTH_ADMIN`
- `android.permission.CHANGE_WIFI_MULTICAST_STATE`
- `android.permission.CHANGE_WIFI_STATE`
- `android.permission.CHANGE_WIMAX_STATE`
- `android.permission.DISABLE_KEYGUARD`
- `android.permission.INTERNET`
- `android.permission.NFC`

5.3 将内部账户添加到账户管理器

账户管理器是 Android OS 的系统，它集中管理帐户信息，是应用访问在线服务和认证令牌所必需的（帐户名称，密码）。 用户需要提前将账户信息注册到账户管理器，当应用尝试访问在线服务时，账户管理器在获得用户权限后，会自动提供应用认证令牌。 账户管理器的优势在于，应用不需要处理极其敏感的信息和密码。

使用账户管理器 的账户管理功能的结构如下图 5.3-1 所示。 “请求应用”是通过获取认证令牌，访问在线服务的应用，这是上述应用。 另一方面，“认证器应用”是账户管理器的功能扩展，并且向账户管理器提供称为认证器的对象，以便账户管理器可集中管理在线服务的账户信息和认证令牌。 请求应用和认证器应用不需要是单独的应用，因此这些应用可以实现为单个应用。

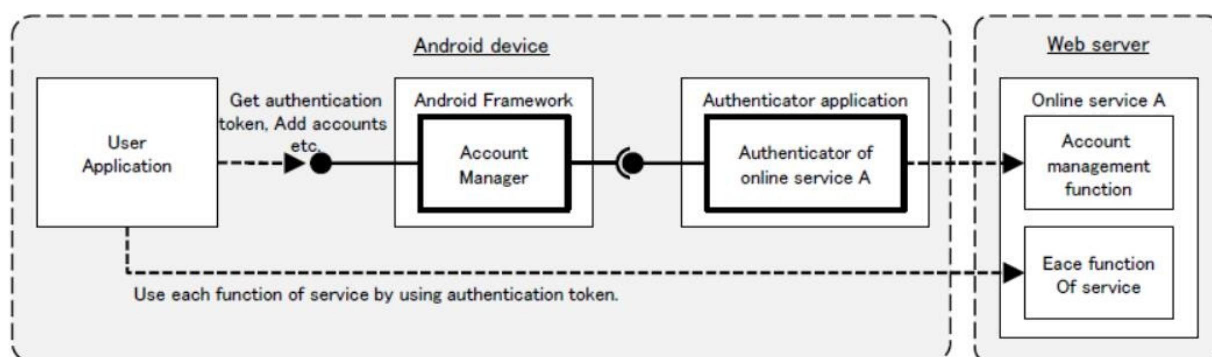


Figure 5.3-1Configuration of account management function which uses Account Manager

最初，用户应用（请求应用）和认证器应用的开发人员签名密钥可以是不同的密钥。 但是，Android 框架的错误仅在 Android 4.0.x 设备中存在 ，并且当用户应用和认证期应用的签名密钥不同时，用户应用中会发生异常，并且不能使用内部账户。 以下示例代码没有针对此缺陷实现任何替代方式。 详细信息请参阅“5.3.3.2 在 Android 4.0.x 中，用户应用和认证程序的签名密钥不同时发生的异常”。

5.3.1 示例代码

“5.3.1.1 创建内部帐户”是认证器应用的示例，“5.3.1.2 使用内部帐户”是请求应用的示例。 在 JSSEC 网站上分发的示例代码集中，每个代码集都对应账户管理器的认证器和用户。

5.3.1.1 创建内部账户

以下是认证器应用的示例代码，它使账户管理器能够使用内部帐户。 在此应用中没有可以从主屏幕启动的活动。 请注意，它间接通过账户管理器，从另一个示例代码“5.3.1.2 使用内部帐户”调用。

要点：

1. 提供认证器的服务必须是私有的。
2. 登录界面的活动必须在验证器应用中实现。
3. 登录界面的活动必须实现为公共活动。
4. 指定登录界面的活动的类名的显式意图，必须设置为 `KEY_INTENT` 。
5. 敏感信息（如帐户信息或认证令牌）不得输出到日志中。
6. 密码不应保存在帐户管理器中。
7. HTTPS 应该用于认证器与在线服务之间的通信。

提供认证器的账户管理器 IBinder 的服务，在 `AndroidManifest.xml` 中定义。 通过元数据指定编写认证器的资源XML文件。

账户管理器认证器/AndroidManifest.xml

通过 XML 文件定义认证器，指定内部账户的账户类型以及其他。

```
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.accountmanager.authenticator"
    xmlns:tools="http://schemas.android.com/tools">
    <!-- Necessary Permission to implement Authenticator -->
    <uses-permission android:name="android.permission.GET_ACCOUN
TS" />
    <uses-permission android:name="android.permission.AUTHENTICA
TE_ACCOUNTS" />
    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <!-- Service which gives IBinder of Authenticator to Acc
ountManager -->
        <!-- *** POINT 1 *** The service that provides an authen
ticator must be private. -->
        <service
            android:name=".AuthenticationService"
            android:exported="false" >
            <!-- intent-filter and meta-data are usual pattern.
-->
            <intent-filter>
                <action android:name="android.accounts.AccountAu
thenticator" />
            </intent-filter>
            <meta-data
            android:name="android.accounts.AccountAuthenticator"
            android:resource="@xml/authenticator" />
        </service>
        <!-- Activity for for login screen which is displayed wh
en adding an account -->
        <!-- *** POINT 2 *** The login screen activity must be i
mplemented in an authenticator applicati
        on. -->
        <!-- *** POINT 3 *** The login screen activity must be m
ade as a public activity. -->
        <activity
            android:name=".LoginActivity"
            android:exported="true"
            android:label="@string/login_activity_title"
            android:theme="@android:style/Theme.Dialog"
            tools:ignore="ExportedActivity" />
    </application>
</manifest>
```

通过 XML 文件定义认证器，指定内部账户的账户类型以及其他。

res/xml/authenticator.xml

```
<account-authenticator xmlns:android="http://schemas.android.com
/apk/res/android"
    android:accountType="org.jssec.android.accountmanager"
    android:icon="@drawable/ic_launcher"
    android:label="@string/label"
    android:smallIcon="@drawable/ic_launcher"
    android:customTokens="true" />
```

为 `AccountManager` 提供 `Authenticator` 实例的服务。 简单的实现返回 `JssecAuthenticator` 类的实例，它就是由 `onBind()` 在此示例中实现的 `Authenticator` ，这就足够了。

AuthenticationService.java

```
package org.jssec.android.accountmanager.authenticator;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

public class AuthenticationService extends Service {

    private JssecAuthenticator mAuthenticator;

    @Override
    public void onCreate() {
        mAuthenticator = new JssecAuthenticator(this);
    }

    @Override
    public IBinder onBind(Intent intent) {
        return mAuthenticator.getIBinder();
    }
}
```

`JssecAuthenticator` 是在此示例中实现的认证器。 它继承了 `AbstractAccountAuthenticator` ，并且实现了所有的抽象方法。 这些方法由账户管理器调用。 在 `addAccount()` 和 `getAuthToken()` 中，用于启动 `LoginActivity` ，从在线服务中获取认证令牌的意图返回到账户管理器。

JssecAuthenticator.java

```
package org.jssec.android.accountmanager.authenticator;

import android.accounts.AbstractAccountAuthenticator;
import android.accounts.Account;
import android.accounts.AccountAuthenticatorResponse;
import android.accounts.AccountManager;
```

```
import android.accounts.NetworkErrorException;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;

public class JssecAuthenticator extends AbstractAccountAuthentic
ator {

    public static final String JSSEC_ACCOUNT_TYPE = "org.jssec.a
ndroid.accountmanager";
    public static final String JSSEC_AUTHTOKEN_TYPE = "webservic
e";
    public static final String JSSEC_AUTHTOKEN_LABEL = "JSSEC We
b Service";
    public static final String RE_AUTH_NAME = "reauth_name";
    protected final Context mContext;

    public JssecAuthenticator(Context context) {
        super(context);
        mContext = context;
    }

    @Override
    public Bundle addAccount(AccountAuthenticatorResponse respon
se, String accountType,
        String authTokenType, String[] requiredFeatures, Bundle
options)
        throws NetworkErrorException {
        AccountManager am = AccountManager.get(mContext);
        Account[] accounts = am.getAccountsByType(JSSEC_ACCOUNT_
TYPE);
        Bundle bundle = new Bundle();
        if (accounts.length > 0) {
            // In this sample code, when an account already exis
ts, consider it as an error.
            bundle.putString(AccountManager.KEY_ERROR_CODE, Stri
ng.valueOf(-1));
            bundle.putString(AccountManager.KEY_ERROR_MESSAGE,
            mContext.getString(R.string.error_account_exists));
        } else {
            // *** POINT 2 *** The login screen activity must be
 implemented in an authenticator application.
            // *** POINT 4 *** The explicit intent which the cla
ss name of the login screen activity is specified must be set to
 KEY_INTENT.
            Intent intent = new Intent(mContext, LoginActivity.c
lass);
            intent.putExtra(AccountManager.KEY_ACCOUNT_AUTHENTIC
ATOR_RESPONSE, response);
            bundle.putParcelable(AccountManager.KEY_INTENT, inte
nt);
        }
        return bundle;
```

```
    }

    @Override
    public Bundle getAuthToken(AccountAuthenticatorResponse resp
onse, Account account,
            String authTokenType, Bundle options) throws NetworkErro
rException {
        Bundle bundle = new Bundle();
        if (accountExist(account)) {
            // *** POINT 4 *** KEY_INTENT must be given an expli
cit intent that is specified the class name of the login screen
activity.
            Intent intent = new Intent(mContext, LoginActivity.c
lass);
            intent.putExtra(RE_AUTH_NAME, account.name);
            bundle.putParcelable(AccountManager.KEY_INTENT, inte
nt);
        } else {
            // When the specified account doesn't exist, conside
r it as an error.
            bundle.putString(AccountManager.KEY_ERROR_CODE, Stri
ng.valueOf(-2));
            bundle.putString(AccountManager.KEY_ERROR_MESSAGE,
            mContext.getString(R.string.error_account_not_exists
));
        }
        return bundle;
    }

    @Override
    public String getAuthTokenLabel(String authTokenType) {
        return JSSEC_AUTHTOKEN_LABEL;
    }

    @Override
    public Bundle confirmCredentials(AccountAuthenticatorRespons
e response, Account account,
            Bundle options) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle editProperties(AccountAuthenticatorResponse re
sponse, String accountType) {
        return null;
    }

    @Override
    public Bundle updateCredentials(AccountAuthenticatorResponse
 response, Account account,
            String authTokenType, Bundle options) throws NetworkErro
rException {
        return null;
```

```
    }

    @Override
    public Bundle hasFeatures(AccountAuthenticatorResponse respo
nse, Account account,
        String[] features) throws NetworkErrorException {
        Bundle result = new Bundle();
        result.putBoolean(AccountManager.KEY_BOOLEAN_RESULT, fal
se);
        return result;
    }

    private boolean accountExist(Account account) {
        AccountManager am = AccountManager.get(mContext);
        Account[] accounts = am.getAccountsByType(JSSEC_ACCOUNT_
TYPE);
        for (Account ac : accounts) {
            if (ac.equals(account)) {
                return true;
            }
        }
        return false;
    }
}
```

这是登录活动，它向在线服务发送帐户名称和密码，并执行登录认证，并因此获得认证令牌。 它会在添加新帐户或再次获取认证令牌时显示。 假设在线服务的实际访问在 `WebService` 类中实现。

LoginActivity.java

```
package org.jssec.android.accountmanager.authenticator;

import org.jssec.android.accountmanager.webservice.WebService;
import android.accounts.Account;
import android.accounts.AccountAuthenticatorActivity;
import android.accounts.AccountManager;
import android.content.Intent;
import android.os.Bundle;
import android.text.InputType;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.view.Window;
import android.widget.EditText;

public class LoginActivity extends AccountAuthenticatorActivity
{

    private static final String TAG = AccountAuthenticatorActivi
ty.class.getSimpleName();
```

```
    private String mReAuthName = null;
    private EditText mNameEdit = null;
    private EditText mPassEdit = null;

    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        // Display alert icon
        requestWindowFeature(Window.FEATURE_LEFT_ICON);
        setContentView(R.layout.login_activity);
        getWindow().setFeatureDrawableResource(Window.FEATURE_LE
FT_ICON,
        android.R.drawable.ic_dialog_alert);
        // Find a widget in advance
        mNameEdit = (EditText) findViewById(R.id.username_edit);
        mPassEdit = (EditText) findViewById(R.id.password_edit);
        // *** POINT 3 *** The login screen activity must be mad
e as a public activity, and suppose the attack access from other
 application.
        // Regarding external input, only RE_AUTH_NAME which is
String type of Intent#extras, are handled.
        // This external input String is passed toextEdit#setTex
t(), WebService#login(),new Account(),
        // as a parameter,it's verified that there's no problem
if any character string is passed.
        mReAuthName = getIntent().getStringExtra(JssecAuthentica
tor.RE_AUTH_NAME);
        if (mReAuthName != null) {
        // Since LoginActivity is called with the specified user
 name, user name should not be editable.
        mNameEdit.setText(mReAuthName);
        mNameEdit.setInputType(InputType.TYPE_NULL);
        mNameEdit.setFocusable(false);
        mNameEdit.setEnabled(false);
        }
    }

    // It's executed when login button is pressed.
    public void handleLogin(View view) {
        String name = mNameEdit.getText().toString();
        String pass = mPassEdit.getText().toString();
        if (TextUtils.isEmpty(name) || TextUtils.isEmpty(pass))
{
            // Process when the inputed value is incorrect
            setResult(RESULT_CANCELED);
            finish();
        }
        // Login to online service based on the inpputed accoun
t information.
        WebService web = new WebService();
        String authToken = web.login(name, pass);
        if (TextUtils.isEmpty(authToken)) {
            // Process when authentication failed
```

```
            setResult(RESULT_CANCELED);
            finish();
        }
        // Process when login was successful, is as per below.
        // *** POINT 5 *** Sensitive information (like account i
nformation or authentication token) must not be output to the lo
g.
        Log.i(TAG, "WebService login succeeded");
        if (mReAuthName == null) {
            // Register accounts which logged in successfully, t
o aAccountManager
            // *** POINT 6 *** Password should not be saved in A
ccount Manager.
            AccountManager am = AccountManager.get(this);
            Account account = new Account(name, JssecAuthenticat
or.JSSEC_ACCOUNT_TYPE);
            am.addAccountExplicitly(account, null, null);
            am.setAuthToken(account, JssecAuthenticator.JSSEC_AU
THTOKEN_TYPE, authToken);
            Intent intent = new Intent();
            intent.putExtra(AccountManager.KEY_ACCOUNT_NAME, nam
e);
            intent.putExtra(AccountManager.KEY_ACCOUNT_TYPE,
                JssecAuthenticator.JSSEC_ACCOUNT_TYPE);
            setAccountAuthenticatorResult(intent.getExtras());
            setResult(RESULT_OK, intent);
        } else {
            // Return authentication token
            Bundle bundle = new Bundle();
            bundle.putString(AccountManager.KEY_ACCOUNT_NAME, na
me);
            bundle.putString(AccountManager.KEY_ACCOUNT_TYPE,
                JssecAuthenticator.JSSEC_ACCOUNT_TYPE);
            bundle.putString(AccountManager.KEY_AUTHTOKEN, authT
oken);
            setAccountAuthenticatorResult(bundle);
            setResult(RESULT_OK);
        }
        finish();
    }
}
```

实际上，`WebService` 类在这里是虚拟实现，这是假设认证总是成功的示例实现，并且固定字符串作为认证令牌返回。

WebService.java

```
package org.jssec.android.accountmanager.webservice;
public class WebService {

    /**
    * Suppose to access to account managemnet function of online
 service.
    *
    * @param username Account name character string
    * @param password password character string
    * @return Return authentication token
    */
    public String login(String username, String password) {
        // *** POINT 7 *** HTTPS should be used for communicatio
n between an authenticator and the online services.
        // Actually, communication process with servers is imple
mented here, but Omit here, since this is a sample.
        return getAuthToken(username, password);
    }

    private String getAuthToken(String username, String password)
 {
        // In fact, get the value which uniqueness and impossibi
lity of speculation are guaranteed by the server,
        // but the fixed value is returned without communication
 here, since this is sample.
        return "c2f981bda5f34f90c0419e171f60f45c";
    }
}
```

5.3.1.2 使用内部账户

以下是应用示例代码，它添加内部帐户并获取认证令牌。 当另一个示例应用“5.3.1.1 创建内部帐户”安装在设备上时，可以添加内部帐户或获取认证令牌。 仅当两个应用的签名密钥不同时，才会显示“访问请求”界面。

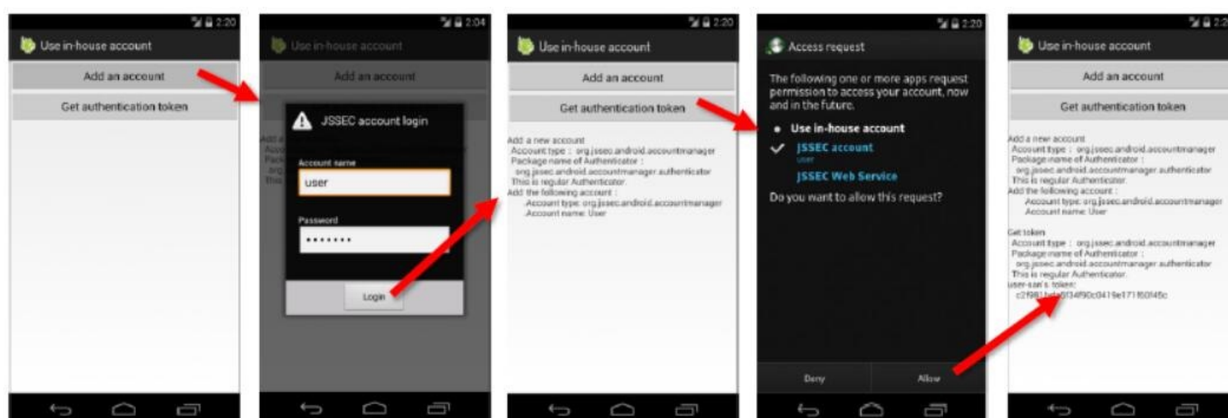


Figure 5.3-2 Behavior screen of sample application AccountManager User

要点：

在验证认证器是否正常之后，执行账户流程。

`AccountManager` 用户应用的 `AndroidManifest.xml` 。声明使用必要的权限。请参阅“5.3.3.1 账户管理器和权限的使用”来了解必要的权限。

账户管理器用户/AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.accountmanager.user" >
    <uses-permission android:name="android.permission.GET_ACCOUN
TS" />
    <uses-permission android:name="android.permission.MANAGE_ACC
OUNTS" />
    <uses-permission android:name="android.permission.USE_CREDEN
TIALS" />
    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".UserActivity"
            android:label="@string/app_name"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

用户应用的活动。当点击屏幕上的按钮时，会执行 `addAccount()` 或 `getAuthToken()` 。在某些情况下，对应特定帐户类型的认证器可能是伪造的，因此请注意在验证认证器正常后，启动帐户流程。

UserActivity.java

```
package org.jssec.android.accountmanager.user;

import java.io.IOException;
import org.jssec.android.shared.PkgCert;
import org.jssec.android.shared.Utils;
import android.accounts.Account;
import android.accounts.AccountManager;
import android.accounts.AccountManagerCallback;
import android.accounts.AccountManagerFuture;
```

```
import android.accounts.AuthenticatorDescription;
import android.accounts.AuthenticatorException;
import android.accounts.OperationCanceledException;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class UserActivity extends Activity {

    // Information of the Authenticator to be used
    private static final String JSSEC_ACCOUNT_TYPE = "org.jssec.
android.accountmanager";
    private static final String JSSEC_TOKEN_TYPE = "webservice";
    private TextView mLogView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.user_activity);
        mLogView = (TextView)findViewById(R.id.logview);
    }

    public void addAccount(View view) {
        logLine();
        logLine("Add a new account");
        // *** POINT 1 *** Execute the account process after ver
ifying if the authenticator is regular one.
        if (!checkAuthenticator()) return;
        AccountManager am = AccountManager.get(this);
        am.addAccount(JSSEC_ACCOUNT_TYPE, JSSEC_TOKEN_TYPE, null
, null, this,
            new AccountManagerCallback<Bundle>() {

                @Override
                public void run(AccountManagerFuture<Bundle> fut
ure) {
                    try {
                        Bundle result = future.getResult();
                        String type = result.getString(AccountMa
nager.KEY_ACCOUNT_TYPE);
                        String name = result.getString(AccountMa
nager.KEY_ACCOUNT_NAME);
                        if (type != null && name != null) {
                            logLine("Add the following accounts:"
);
                            logLine(" Account type: %s", type);
                            logLine(" Account name: %s", name);
                        } else {
                            String code = result.getString(Accou
ntManager.KEY_ERROR_CODE);
                            String msg = result.getString(Accoun
```

```
tManager.KEY_ERROR_MESSAGE);
                            logLine("The account cannot be added"
);
                            logLine(" Error code %s: %s", code,
msg);
                        }
                    } catch (OperationCanceledException e) {
                    } catch (AuthenticatorException e) {
                    } catch (IOException e) {
                    }
                }
            }, null);
    }

    public void getAuthToken(View view) {
        logLine();
        logLine("Get token");
        // *** POINT 1 *** After checking that the Authenticator
 is the regular one, execute account process.
        if (!checkAuthenticator()) return;
        AccountManager am = AccountManager.get(this);
        Account[] accounts = am.getAccountsByType(JSSEC_ACCOUNT_
TYPE);
        if (accounts.length > 0) {
            Account account = accounts[0];
            am.getAuthToken(account, JSSEC_TOKEN_TYPE, null, this
,
                new AccountManagerCallback<Bundle>() {

                    @Override
                    public void run(AccountManagerFuture<Bundle>
 future) {
                        try {
                            Bundle result = future.getResult();
                            String name = result.getString(Accou
ntManager.KEY_ACCOUNT_NAME);
                            String authtoken = result.getString(
AccountManager.KEY_AUTHTOKEN);
                            logLine("%s-san's token:", name);
                            if (authtoken != null) {
                                logLine(" %s", authtoken);
                            } else {
                                logLine(" Couldn't get");
                            }
                        } catch (OperationCanceledException e) {
                            logLine(" Exception: %s",e.getClass(
).getName());
                        } catch (AuthenticatorException e) {
                            logLine(" Exception: %s",e.getClass(
).getName());
                        } catch (IOException e) {
                            logLine(" Exception: %s",e.getClass(
).getName());
```

```
                        }
                    }
                }, null);
            } else {
                logLine("Account is not registered.");
            }
        }

    // *** POINT 1 *** Verify that Authenticator is regular one.
    private boolean checkAuthenticator() {
        AccountManager am = AccountManager.get(this);
        String pkgname = null;
        for (AuthenticatorDescription ad : am.getAuthenticatorTy
pes()) {
            if (JSSEC_ACCOUNT_TYPE.equals(ad.type)) {
                pkgname = ad.packageName;
                break;
            }
        }
        if (pkgname == null) {
            logLine("Authenticator cannot be found.");
            return false;
        }
        logLine(" Account type: %s", JSSEC_ACCOUNT_TYPE);
        logLine(" Package name of Authenticator: ");
        logLine(" %s", pkgname);
        if (!PkgCert.test(this, pkgname, getTrustedCertificateHa
sh(this))) {
            logLine(" It's not regular Authenticator(certificate
 is not matched.)");
            return false;
        }
        logLine(" This is regular Authenticator.");
        return true;
    }

    // Certificate hash value of regular Authenticator applicati
on
    // Certificate hash value can be checked in sample applciati
on JSSEC CertHash Checker
    private String getTrustedCertificateHash(Context context) {
        if (Utils.isDebuggable(context)) {
            // Certificate hash value of debug.keystore "android
debugkey"
            return "0EFB7236 328348A9 89718BAD DF57F544 D5CCB4AE
 B9DB34BC 1E29DD26 F77C8255";
        } else {
            // Certificate hash value of keystore "my company ke
y"
            return "D397D343 A5CBC10F 4EDDEB7C A10062DE 5690984F
 1FB9E88B D7B3A7C2 42E142CA";
        }
    }
```

```
    private void log(String str) {
        mLogView.append(str);
    }

    private void logLine(String line) {
        log(line + "¥n");
    }

    private void logLine(String fmt, Object... args) {
        logLine(String.format(fmt, args));
    }

    private void logLine() {
        log("¥n");
    }
}
```

PkgCert.java

```
package org.jssec.android.shared;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;

public class PkgCert {

    public static boolean test(Context ctx, String pkgname, Stri
ng correctHash) {
        if (correctHash == null) return false;
        correctHash = correctHash.replaceAll(" ", "");
        return correctHash.equals(hash(ctx, pkgname));
    }

    public static String hash(Context ctx, String pkgname) {
        if (pkgname == null) return null;
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pkginfo = pm.getPackageInfo(pkgname, Pac
kageManager.GET_SIGNATURES);
            if (pkginfo.signatures.length != 1) return null; //
Will not handle multiple signatures.
            Signature sig = pkginfo.signatures[0];
            byte[] cert = sig.toByteArray();
            byte[] sha256 = computeSha256(cert);
```

```
            return byte2hex(sha256);
        } catch (NameNotFoundException e) {
            return null;
        }
    }

    private static byte[] computeSha256(byte[] data) {
        try {
            return MessageDigest.getInstance("SHA-256").digest(d
ata);
        } catch (NoSuchAlgorithmException e) {
            return null;
        }
    }

    private static String byte2hex(byte[] data) {
        if (data == null) return null;
        final StringBuilder hexadecimal = new StringBuilder();
        for (final byte b : data) {
            hexadecimal.append(String.format("%02X", b));
        }
        return hexadecimal.toString();
    }
}
```

5.3.2 规则书

实现认证器应用时，遵循下列规则：

5.3.2.1 提供认证器的服务必须是私有的（必需）

前提是，提供认证器的服务由账户管理器使用，并且不应该被其他应用访问。 因此，通过使其成为私有服务，它可以避免其他应用的访问。 此外，账户管理器以系统权限运行，所以即使是私有服务，账户管理器也可以访问。

5.3.2.2 登录界面活动必须由认证器应用实现（必需）

用于添加新帐户并获取认证令牌的登录界面，应由认证应用实现。 自己的登录界面不应该在用户应用一端准备。 正如本文开头提到的，【账户管理器的优势在于，极其敏感的信息/密码不一定要由应用处理】，如果在用户应用一端准备登录界面，则密码由用户应用处理， 其设计越过了账户管理器的策略。

通过由身份验证器应用准备登录界面，操作登录界面的人仅限于设备用户。 这意味着，恶意应用无法通过尝试直接登录，或创建帐户来攻击帐户。

5.3.2.3 登录界面活动必须是公共活动，并假设其他应用的攻击访问（必需）

登录界面活动是由用户应用加载的系统。 为了即使在用户应用和身份验证器应用的签名密钥不同时，也能展示登录界面，登录界面活动应该实现为公共活动。 登录界面活动是公共活动，意味着有可能会被恶意应用启动。 永远不要相信任何输入数据。 因此，有必要采取“3.2 小心并安全处理输入数据”中提到的对策。

5.3.2.4 使用显示意图提供 `KEY_INTENT` ，带有登录界面活动的指定类名称（必需）

当认证器需要打开登录界面活动时，启动登录界面活动的意图，会在返回给账户管理器的 Bundle 中，由 `KEY_INTENT` 提供。 所提供的意图应该是指定登录界面活动的类名的显式意图。 在使用隐示意图，它指定动作名称的情况下，有可能并不启动由认证器应用本身准备的登录界面活动，而是其他应用准备的活动。 当恶意应用准备了和常规一样的登录界面时，用户可能会在伪造的登录界面中输入密码。

5.3.2.5 敏感信息（如帐户信息和认证令牌）不得输出到日志（必需）

访问在线服务的应用有时会遇到麻烦，例如无法成功访问在线服务。 访问失败的原因各不相同，如网络环境管理不善，通信协议实现失败，权限不足，认证错误等。 一个常见的实现方式是，程序输出详细信息给日志，以便开发人员可以稍后分析问题的原因。

敏感信息（如密码或认证令牌）不应输出到日志中。 日志信息可以从其他应用读取，因此可能成为信息泄露的原因。 此外，如果帐户名称的泄漏可能导致损失，则不应将帐户名称输出到日志中。

5.3.2.6 密码不应该保存在账户管理器中（推荐）

两个认证信息，密码和认证令牌可以保存在一个账户中，来注册账户管理器。 这些信息将以明文形式（即不加密）存储在以下目录下的`accounts.db`中。

- Android 4.1 及之前：`/data/system/accounts.db`
- Android 4.2 及之后：`/data/system/0/accounts.db or /data/system/<UserId>/accounts`

要阅读`accounts.db`的内容，需要 root 权限或系统权限，并且无法从市场上的 Android 设备中读取它。 在 Android 操作系统中存在漏洞的情况下，攻击者可以获得 root 权限或系统权限，保存在`accounts.db`中的认证信息将处在风险边缘。

本文中介绍的认证应用旨在将认证令牌保存在账户管理器中，而不保存用户密码。 在一定时间内连续访问在线服务时，通常认证令牌的有效期限会延长，因此在大多数情况下，不保存密码的设计就足够了。

通常，认证令牌的有效期限比密码短，并且它的特点是可以随时禁用。 如果认证令牌泄漏，则可以将其禁用，因此与密码相比，认证令牌比较安全。 在认证令牌被禁用的情况下，用户可以再次输入密码以获得新的认证令牌。

如果在密码泄漏时禁用密码，用户将无法再使用在线服务。 在这种情况下，它需要呼叫中心支持等，这将花费巨大的成本。 因此，最好从设计中避免在账户管理器中保存密码。 在不能避免保存密码的设计的情况下，应该采取高级别的逆向工程对策，如加密密码和混淆加密密钥。

5.3.2.7 HTTPS 应该用于认证器和在线服务之间的通信（必需）

密码或认证令牌就是所谓的认证信息，如果被第三方接管，第三方可以伪装成有效用户。 由于认证器使用在线服务来发送/接收这些类型的认证信息，因此应使用可靠的加密通信方法，如 HTTPS。

5.3.2.8 应该在验证认证器是否正常之后，执行帐户流程（必需）

如果有多个认证器在设备中定义了相同的帐户类型，则先前安装的认证器将生效。 所以，安装自己的认证器之后，它不会被使用。

如果之前安装的认证器是恶意软件的伪装，则用户输入的帐户信息可能被恶意软件接管。 在执行帐户操作之前，用户应用应验证执行帐户操作的帐户类型，不管是否分配了常规认证器。

可以通过检查认证器的包的证书散列值，是否匹配预先确认的有效证书散列值，来验证分配给账户类型的认证器是否是正常的。 如果发现证书哈希值不匹配，则最好提示用户卸载程序包，它包含分配给该帐户类型的意外的认证验证器。

5.3.3 高级话题

5.3.3.1 账户管理和权限的使用

要使用`AccountManager`类的每种方法，都需要在应用的`AndroidManifest.xml`中分别声明使用相应的权限。 表 5.3-1 显示了权限和方法的对应关系。

表 5.3-1 账户管理器的函数以及权限

	账户管理器提供的函数
权限	方法
`AUTHENTICATE_ACCOUNTS`（只有由认证器的相同密钥签名的软件包才可以使用。）	`getPassword()`
	`getUserData()`
	`addAccountExplicitly()`
	`peekAuthToken()`
	`setAuthToken()`

	`setPassword()`
	`setUserData()`
	`renameAccount()`
`GET_ACCOUNTS`	`getAccounts()`
	`getAccountsByType()`
	`getAccountsByTypeAndFeature`

	`addOnAccountsUpdatedListene`
	`hasFeatures()`
`MANAGE_ACCOUNTS`	`getAuthTokenByFeatures()`
	`addAccount()`
	`removeAccount()`

	`removeAccount()`
	`clearPassword()`
	`updateCredentials()`
	`editProperties()`
	`confirmCredentials()`
`USE_CREDENTIALS`	`getAuthToken()`
	`blockingGetAuthToken()`

MANAGE_ACCOUNTS 或 USE_CREDENTIALS	invalidateAuthToken()

在使用需要 AUTHENTICATE_ACCOUNTS 权限的方法组的情况下，存在软件包的签名密钥以及权限相关的限制。 具体来说，提供认证器的包的签名密钥，和使用方法的应用的包的签名密钥应该是相同的。 因此，在分发使用方法组的应用时，除了认证器之外，必须使用 AUTHENTICATE_ACCOUNTS 权限，并且应使用认证器的相同密钥进行签名。

在 Android Studio 的开发阶段，由于固定的调试密钥库可能会被某些 Android Studio 项目共享，开发人员可能只考虑权限而不考虑签名，来实现和测试帐户管理器。 特别是，对于对每个应用使用不同签名密钥的开发人员来说，因为这种限制，在选择用于应用的密钥时要非常小心。 此外，由于 AccountManager 获得的数据包含敏感信息，因此需要小心处理，来减少泄漏或未授权使用的风险。

5.3.3.2 在 Android 4.0.x 中，用户应用和认证器应用的签名密钥不同时发生的异常

认证令牌获取功能是由开发者密钥签发的用户应用所需的，它不同于认证器应用的签名密钥。通过显示 认证令牌许可证屏幕（ GrantCredentialsPermissionActivity ）， AccountManager 验证用户是否授予认证令牌的使用权。但是 Android 4.0.x 的 Android 框架中存在一个错误，只要 AccountManager 打开此屏幕，就会发生异常并且应用被强制关闭 。（图 5.3-3）。 错误的详细信息，请参阅 https://code.google.com/p/android/issues/detail?id=23421。 这个 bug 在 Android 4.1.x 及更高版本中无法找到。

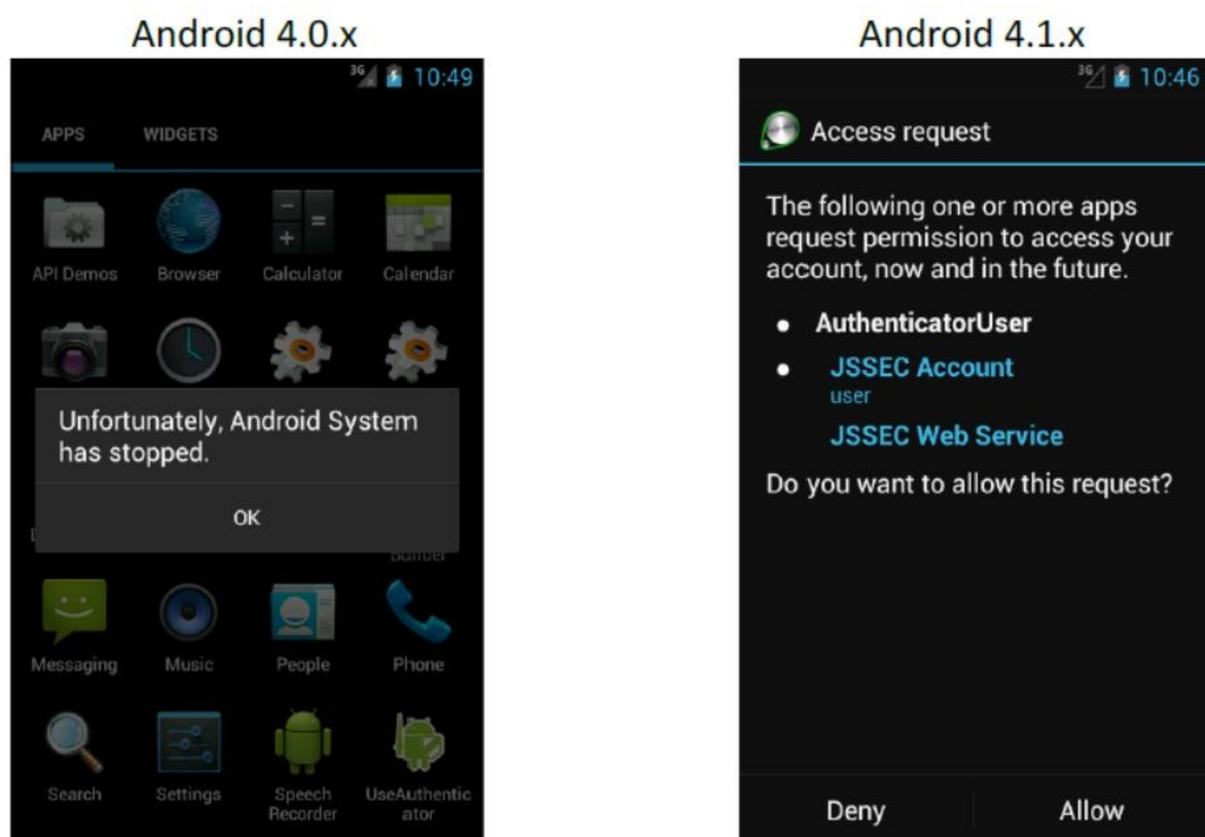


Figure 5.3-3 When displaying Android standard authentication token license screen.

5.4 通过 HTTPS 的通信

大多数智能手机应用都与互联网上的 Web 服务器通信。 作为通信的方法，我们在这里集中讨论 HTTP 和 HTTPS 的两种方法。 从安全角度来看，HTTPS 通信更为可取。 最近，Google 或 Facebook 等主要 Web 服务已经开始使用 HTTPS 作为默认设置。

自 2012 年以来，Android 应用中 HTTPS 通信实现的许多缺陷已被指出。 这些缺陷可能用于访问由服务器证书操作的测试 Web 服务器，这些服务器证书不是由可信的第三方证书机构颁发，而是由私人（以下称为私有证书）颁发。

本节将解释 HTTP 和 HTTPS 通信方法，并介绍使用 HTTPS 安全地访问由私有证书操作的 Web 服务器的方法。

5.4.1 示例代码

你可以通过下面的图表（图 5.4-1）找出你应该实现的 HTTP / HTTPS 通信类型。

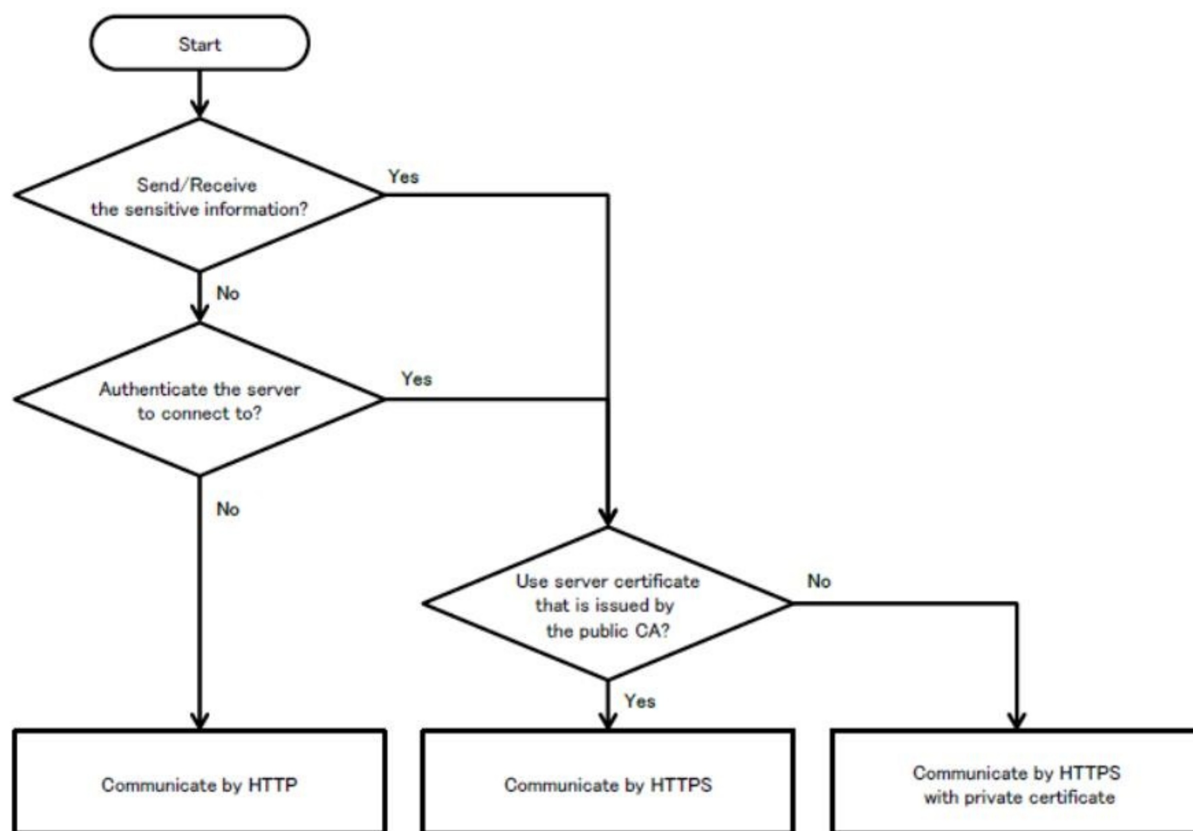


Figure 5.4-1 Flow Figure to select sample code of HTTP/HTTPS

当发送或接收敏感信息时，将使用 HTTPS 通信，因为其通信通道使用 SSL / TLS 加密。 以下敏感信息需要 HTTPS 通信。

- Web 服务的登录 ID /密码。
- 保持认证状态的信息（会话 ID，令牌，Cookie 等）
- 取决于 Web 服务的重要/机密信息（个人信息，信用卡信息等）

具有网络通信的智能手机应用是“系统”和 Web 服务器的一部分。 而且你必须根据整个“系统”的安全设计和编码，为每个通信选择 HTTP 或 HTTPS。 表 5.4-1 用于比较 HTTP 和 HTTPS。 表 5.4-2 是示例代码的差异。

表 5.4-1 HTTP 与 HTTPS 通信方式的比较

	HTTP	HTTPS	
特性	URL	`http://` 开头	`https://` 开头
	加密内容	否	是
	内容的篡改检测	不可能	可能
	对服务器进行认证	不可能	可能
损害的风险	由攻击者读取内容	高	低
	由攻击者修改内容	高	低
	应用访问了伪造的服务器	高	低

表 5.4-2 HTTP/HTTPS 通信示例代码的解释

示例代码	通信	收发敏感信息	服务器证书
通过 HTTP 的通信	HTTP	不适用	-
通过 HTTPS 的通信	HTTPS	OK	服务器证书由可信第三方机构签署，例如 Cybertrust 和 VeriSign
通过 HTTPS 使用私有证书的通信	HTTTPS	OK	私有证书（经常能在内部服务器或测试服务器上看到的操作）

Android 支持 `java.net.HttpURLConnection` / `javax.net.ssl.HttpsURLConnection` 作为 HTTP / HTTPS 通信 API。 在 Android 6.0（API Level 23）版本中，另一个 HTTP 客户端库 Apache HttpClient 的支持已被删除。

5.4.1.1 通过 HTTP 进行通信

它基于两个前提，即通过 HTTP 通信发送/接收的所有内容都可能被攻击者嗅探和篡改，并且你的目标服务器可能被攻击者准备的假服务器替换。 只有在没有造成损害或损害在允许范围内的情况下，才能使用 HTTP 通信，即使在本地也是如此。 如果应用无法接受该前提，请参阅“5.4.1.2 通过 HTTPS 进行通信”和“5.4.1.3 通过 HTTPS 使用私有证书进行通信”。

以下示例代码显示了一个应用，它在 Web 服务器上执行图像搜索，获取结果图像并显示它。与服务器的 HTTP 通信在搜索时执行两次。第一次通信是搜索图像数据，第二次是获取它。它使用 `AsyncTask` 创建用于通信过程的工作线程，来避免在 UI 线程上执行通信。与服务器的通信中发送/接收的内容，在这里不被认为是敏感的（例如，用于搜索的字符串，图像的 URL 或图像数据）。因此，接收到的数据，如图像的 URL 和图像数据，可能由攻击者提供。为了简单地显示示例代码，

在示例代码中没有采取任何对策，通过将接收到的攻击数据视为可容忍的。此外，在 JSON 解析或显示图像数据期间，可能出现异常的处理将被忽略。根据应用规范，有必要正确处理例外情况。

要点：

1. 发送的数据中不得包含敏感信息。
2. 假设收到的数据可能来自攻击者。

HttpImageSearch.java

```
package org.jssec.android.https.imagesearch;

import android.os.AsyncTask;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.BufferedInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.URL;

public abstract class HttpImageSearch extends AsyncTask<String,
Void, Object> {

    @Override
    protected Object doInBackground(String... params) {
        byte[] responseArray;
        // ----------------------------------------------------------------
---
        // Communication 1st time: Execute image search
        // ----------------------------------------------------------------
---
        // *** POINT 1 *** Sensitive information must not be con
tained in send data.
        // Send image search character string
        StringBuilder s = new StringBuilder();
        for (String param : params){
            s.append(param);
            s.append('+');
        }
        s.deleteCharAt(s.length() - 1);
        String search_url = "http://ajax.googleapis.com/ajax/ser
vices/search/images?v=1.0&q=" +
        s.toString();
        responseArray = getByteArray(search_url);
        if (responseArray == null) {
            return null;
        }
        // *** POINT 2 *** Suppose that received data may be sen
t from attackers.
        // This is sample, so omit the process in case of the se
```

```
arching result is the data from an attacker.
        // This is sample, so omit the exception process in case
 of JSON purse.
        String image_url;
        try {
            String json = new String(responseArray);
            image_url = new JSONObject(json).getJSONObject("resp
onseData")
                .getJSONArray("results").getJSONObject(0).getStr
ing("url");
        } catch(JSONException e) {
            return e;
        }
        // -----------------------------------------------------
---
        // Communication 2nd time: Get images
        // -----------------------------------------------------
---
        // *** POINT 1 *** Sensitive information must not be con
tained in send data.
        if (image_url != null ) {
            responseArray = getByteArray(image_url);
            if (responseArray == null) {
                return null;
            }
        }
        // *** POINT 2 *** Suppose that received data may be sen
t from attackers.
        return responseArray;
    }

    private byte[] getByteArray(String strUrl) {
        byte[] buff = new byte[1024];
        byte[] result = null;
        HttpURLConnection response;
        BufferedInputStream inputStream = null;
        ByteArrayOutputStream responseArray = null;
        int length;
        try {
            URL url = new URL(strUrl);
            response = (HttpURLConnection) url.openConnection();
            response.setRequestMethod("GET");
            response.connect();
            checkResponse(response);
            inputStream = new BufferedInputStream(response.getIn
putStream());
            responseArray = new ByteArrayOutputStream();
            while ((length = inputStream.read(buff)) != -1) {
                if (length > 0) {
                    responseArray.write(buff, 0, length);
                }
            }
            result = responseArray.toByteArray();
```

```
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (inputStream != null) {
                try {
                    inputStream.close();
                } catch (IOException e) {
                    // This is sample, so omit the exception pro
cess
                }
            }
            if (responseArray != null) {
                try {
                    responseArray.close();
                } catch (IOException e) {
                    // This is sample, so omit the exception pro
cess
                }
            }
        }
        return result;
    }

    private void checkResponse(HttpURLConnection response) throws
 IOException {
        int statusCode = response.getResponseCode();
        if (HttpURLConnection.HTTP_OK != statusCode) {
            throw new IOException("HttpStatus: " + statusCode);
        }
    }
}
```

ImageSearchActivity.java

```
package org.jssec.android.https.imagesearch;

import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;

public class ImageSearchActivity extends Activity {

    private EditText mQueryBox;
    private TextView mMsgBox;
```

```
    private ImageView mImgBox;
    private AsyncTask<String, Void, Object> mAsyncTask ;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mQueryBox = (EditText)findViewById(R.id.querybox);
        mMsgBox = (TextView)findViewById(R.id.msgbox);
        mImgBox = (ImageView)findViewById(R.id.imageview);
    }

    @Override
    protected void onPause() {
        // After this, Activity may be deleted, so cancel the as
ynchronization process in advance.
        if (mAsyncTask != null) mAsyncTask.cancel(true);
        super.onPause();
    }

    public void onHttpSearchClick(View view) {
        String query = mQueryBox.getText().toString();
        mMsgBox.setText("HTTP:" + query);
        mImgBox.setImageBitmap(null);
        // Cancel, since the last asynchronous process might not
 have been finished yet.
        if (mAsyncTask != null) mAsyncTask.cancel(true);
        // Since cannot communicate by UI thread, communicate by
 worker thread by AsynchTask.
        mAsyncTask = new HttpImageSearch() {

            @Override
            protected void onPostExecute(Object result) {
                // Process the communication result by UI thread.

                if (result == null) {
                    mMsgBox.append("¥nException occurs¥n");
                } else if (result instanceof Exception) {
                    Exception e = (Exception)result;
                    mMsgBox.append("¥nException occurs¥n" + e.to
String());
                } else {
                    // Exception process when image display is o
mitted here, since it's sample.
                    byte[] data = (byte[])result;
                    Bitmap bmp = BitmapFactory.decodeByteArray(d
ata, 0, data.length);
                    mImgBox.setImageBitmap(bmp);
                }
            }
        }.execute(query);
        // pass search character string and start asynchronous p
rocess
```

```
    }

    public void onHttpsSearchClick(View view) {
        String query = mQueryBox.getText().toString();
        mMsgBox.setText("HTTPS:" + query);
        mImgBox.setImageBitmap(null);
        // Cancel, since the last asynchronous process might not
 have been finished yet.
        if (mAsyncTask != null) mAsyncTask.cancel(true);
        // Since cannot communicate by UI thread, communicate by
 worker thread by AsynchTask.
        mAsyncTask = new HttpsImageSearch() {
            @Override
            protected void onPostExecute(Object result) {
                // Process the communication result by UI thread.

                if (result instanceof Exception) {
                    Exception e = (Exception)result;
                    mMsgBox.append("¥nException occurs¥n" + e.to
String());
                } else {
                    byte[] data = (byte[])result;
                    Bitmap bmp = BitmapFactory.decodeByteArray(d
ata, 0, data.length);
                    mImgBox.setImageBitmap(bmp);
                }
            }
        }.execute(query);
        // pass search character string and start asynchronous p
rocess
    }
}
```

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="org.jssec.android.https.imagesearch"
    android:versionCode="1"
    android:versionName="1.0">
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:allowBackup="false"
        android:label="@string/app_name" >
        <activity
            android:name=".ImageSearchActivity"
            android:label="@string/app_name"
            android:theme="@android:style/Theme.Light"
            android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

5.4.1.2 使用 HTTPS 进行通信

在 HTTPS 通信中，检查服务器是否可信，以及传输的数据是否加密。 为了验证服务器，Android HTTPS 库验证“服务器证书”，它在 HTTPS 事务的握手阶段从服务器传输，其要点如下：

- 服务器证书由可信的第三方证书机构签署
- 服务器证书的期限和其他属性有效
- 服务器的主机名匹配服务器证书的主题字段中的 CN（通用名称）或 SAN（主题备用名称）

如果上述验证失败，则会引发 `SSLException`（服务器证书验证异常）。 这可能意味着中间人攻击或服务器证书缺陷。 你的应用必须根据应用规范，以适当的顺序处理异常。

下一个示例代码用于 HTTPS 通信，它使用可信的第三方证书机构颁发的服务器证书连接到 Web 服务器。 对于使用私有服务器证书的 HTTPS 通信，请参阅“5.4.1.3 通过 HTTPS 使用私有证书进行通信”。

以下示例代码展示了一个应用，它在 Web 服务器上执行图像搜索，获取结果图像并显示它。 与服务器的 HTTPS 通信在搜索时执行两次。 第一次通信是搜索图像数据，第二次是获取它。 它使用 `AsyncTask` 创建用于通信过程的工作线程，来避免在 UI 线程上执行通信。 与服务器的通信中发送/接收的所有内容，在这里被认为是

敏感的（例如，用于搜索的字符串，图像的 URL 或图像数据）。 为了简单地显示示例代码，不会执行针对 `SSLException` 的特殊处理。 根据应用规范，有必要正确处理异常。 另外，下面的示例代码允许使用 SSLv3 进行通信。 通常，我们建议配置远程服务器上的设置来禁用 SSLv3，以避免针对 SSLv3 中的漏洞（称为 POODLE）的攻击。

要点：

1. URI 以 `https://` 开头。
2. 发送数据中可能包含敏感信息。
3. 尽管数据是从通过 HTTPS 连接的服务器发送的，但要小心并安全地处理收到的数据。
4. `SSLException` 应该在应用中以适当的顺序处理。

HttpsImageSearch.java

```
package org.jssec.android.https.imagesearch;

import org.json.JSONException;
import org.json.JSONObject;
import android.os.AsyncTask;
import java.io.BufferedInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.URL;

public abstract class HttpsImageSearch extends AsyncTask<String,
Void, Object> {

    @Override
    protected Object doInBackground(String... params) {
        byte[] responseArray;
        // -----------------------------------------------------
---
        // Communication 1st time : Execute image search
        // -----------------------------------------------------
---
        // *** POINT 1 *** URI starts with https://.
        // *** POINT 2 *** Sensitive information may be containe
d in send data.
        StringBuilder s = new StringBuilder();
        for (String param : params){
            s.append(param);
            s.append('+');
        }
        s.deleteCharAt(s.length() - 1);
        String search_url = "https://ajax.googleapis.com/ajax/se
rvices/search/images?v=1.0&q=" +
        s.toString();
        responseArray = getByteArray(search_url);
        if (responseArray == null) {
```

```
                return null;
            }
            // *** POINT 3 *** Handle the received data carefully an
d securely,
            // even though the data was sent from the server connect
ed by HTTPS.
            // Omitted, since this is a sample. Please refer to "3.2
 Handling Input Data Carefully and Securely."
            String image_url;
            try {
                String json = new String(responseArray);
                image_url = new JSONObject(json).getJSONObject("resp
onseData")
                    .getJSONArray("results").getJSONObject(0).getStr
ing("url");
            } catch(JSONException e) {
                return e;
            }
            // -----------------------------------------------------
---
            // Communication 2nd time : Get image
            // -----------------------------------------------------
---
            // *** POINT 1 *** URI starts with https://.
            // *** POINT 2 *** Sensitive information may be containe
d in send data.
            if (image_url != null ) {
                responseArray = getByteArray(image_url);
                if (responseArray == null) {
                    return null;
                }
            }
            return responseArray;
        }

        private byte[] getByteArray(String strUrl) {
            byte[] buff = new byte[1024];
            byte[] result = null;
            HttpURLConnection response;
            BufferedInputStream inputStream = null;
            ByteArrayOutputStream responseArray = null;
            int length;
            try {
                URL url = new URL(strUrl);
                response = (HttpURLConnection) url.openConnection();
                response.setRequestMethod("GET");
                response.connect();
                checkResponse(response);
                inputStream = new BufferedInputStream(response.getIn
putStream());
                responseArray = new ByteArrayOutputStream();
                while ((length = inputStream.read(buff)) != -1) {
                    if (length > 0) {
```

```
                    responseArray.write(buff, 0, length);
                }
            }
            result = responseArray.toByteArray();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (inputStream != null) {
                try {
                    inputStream.close();
                } catch (IOException e) {
                    // This is sample, so omit the exception pro
cess
                }
            }
            if (responseArray != null) {
                try {
                    responseArray.close();
                } catch (IOException e) {
                    // This is sample, so omit the exception pro
cess
                }
            }
        }
        return result;
    }

    private void checkResponse(HttpURLConnection response) throws
 IOException {
        int statusCode = response.getResponseCode();
        if (HttpURLConnection.HTTP_OK != statusCode) {
            throw new IOException("HttpStatus: " + statusCode);
        }
    }
}
```

其他示例代码文件与“5.4.1.1 通过 HTTP 进行通信”相同，因此请参阅“5.4.1.1 通过 HTTP 进行通信”。

5.4.1.3 使用私有证书通过 HTTPS 进行通信

这部分展示了一个 HTTPS 通信的示例代码，其中包含私人颁发的服务器证书（私有证书），但不是可信的第三方机构颁发的服务器证书。请参阅“5.4.3.1 如何创建私有证书并配置服务器”，来创建私有证书机构和私有证书的根证书，并在 Web 服务器中设置 HTTPS。示例程序的资产中包含 `cacert.crt` 文件。它是私有证书机构的根证书文件。

以下示例代码展示了一个应用，在 Web 服务器上获取图像并显示该图像。 HTTPS 用于与服务器的通信。 它使用 `AsyncTask` 创建用于通信过程的工作线程，来避免在 UI 线程上执行通信。 与服务器的通信中发送/接收的所有内容（图像的 URL 和图像数据）都被认为是敏感的。 为了简单地显示示例代码，不会执行针对 `SSLException` 的特殊处理。 根据应用规范，有必要正确处理异常。

要点：

1. 使用私人证书机构的根证书来验证服务器证书。
2. URI 以 `https://` 开头。
3. 发送数据中可能包含敏感信息。
4. 接收的数据可以像服务器一样被信任。
5. `SSLException` 应该在应用中以适当的顺序处理。

PrivateCertificathettpsGet.java

```
package org.jssec.android.https.privatecertificate;

import java.io.BufferedInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.URL;
import java.security.KeyStore;
import java.security.SecureRandom;
import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLException;
import javax.net.ssl.SSLSession;
import javax.net.ssl.TrustManagerFactory;
import android.content.Context;
import android.os.AsyncTask;

public abstract class PrivateCertificateHttpsGet extends AsyncTa
sk<String, Void, Object> {

    private Context mContext;

    public PrivateCertificateHttpsGet(Context context) {
        mContext = context;
    }

    @Override
    protected Object doInBackground(String... params) {
        TrustManagerFactory trustManager;
        BufferedInputStream inputStream = null;
        ByteArrayOutputStream responseArray = null;
        byte[] buff = new byte[1024];
        int length;
        try {
            URL url = new URL(params[0]);
```

```
            // *** POINT 1 *** Verify a server certificate with
the root certificate of a private certificate authority.
            // Set keystore which includes only private certific
ate that is stored in assets, to client.
            KeyStore ks = KeyStoreUtil.getEmptyKeyStore();
            KeyStoreUtil.loadX509Certificate(ks,
                mContext.getResources().getAssets().open("cacert
.crt"));
            // *** POINT 2 *** URI starts with https://.
            // *** POINT 3 *** Sensitive information may be cont
ained in send data.
            trustManager = TrustManagerFactory.getInstance(Trust
ManagerFactory.getDefaultAlgorithm());
            trustManager.init(ks);
            SSLContext sslCon = SSLContext.getInstance("TLS");
            sslCon.init(null, trustManager.getTrustManagers(), n
ew SecureRandom());
            HttpURLConnection con = (HttpURLConnection)url.openC
onnection();
            HttpsURLConnection response = (HttpsURLConnection)co
n;
            response.setDefaultSSLSocketFactory(sslCon.getSocket
Factory());
            response.setSSLSocketFactory(sslCon.getSocketFactory
());
            checkResponse(response);
            // *** POINT 4 *** Received data can be trusted as s
ame as the server.
            inputStream = new BufferedInputStream(response.getIn
putStream());
            responseArray = new ByteArrayOutputStream();
            while ((length = inputStream.read(buff)) != -1) {
                if (length > 0) {
                    responseArray.write(buff, 0, length);
                }
            }
            return responseArray.toByteArray();
        } catch(SSLException e) {
            // *** POINT 5 *** SSLException should be handled wi
th an appropriate sequence in an application.
            // Exception process is omitted here since it's samp
le.
            return e;
        } catch(Exception e) {
            return e;
        } finally {
            if (inputStream != null) {
                try {
                    inputStream.close();
                } catch (Exception e) {
                    // This is sample, so omit the exception pro
cess
                }
```

```
            }
            if (responseArray != null) {
                try {
                    responseArray.close();
                } catch (Exception e) {
                    // This is sample, so omit the exception pro
cess
                }
            }
        }
    }

    private void checkResponse(HttpURLConnection response) throws
 IOException {
        int statusCode = response.getResponseCode();
        if (HttpURLConnection.HTTP_OK != statusCode) {
            throw new IOException("HttpStatus: " + statusCode);
        }
    }
}
```

KeyStoreUtil.java

```
package org.jssec.android.https.privatecertificate;

import java.io.IOException;
import java.io.InputStream;
import java.security.KeyStore;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;
import java.security.cert.Certificate;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.util.Enumeration;

public class KeyStoreUtil {

    public static KeyStore getEmptyKeyStore() throws KeyStoreExc
eption,
        NoSuchAlgorithmException, CertificateException, IOExcept
ion {
        KeyStore ks = KeyStore.getInstance("BKS");
        ks.load(null);
        return ks;
    }

    public static void loadAndroidCAStore(KeyStore ks)
    throws KeyStoreException, NoSuchAlgorithmException,
    CertificateException, IOException {
```

```
        KeyStore aks = KeyStore.getInstance("AndroidCAStore");
        aks.load(null);
        Enumeration<String> aliases = aks.aliases();
        while (aliases.hasMoreElements()) {
            String alias = aliases.nextElement();
            Certificate cert = aks.getCertificate(alias);
            ks.setCertificateEntry(alias, cert);
        }
    }

    public static void loadX509Certificate(KeyStore ks, InputStr
eam is)
        throws CertificateException, KeyStoreException {
        try {
            CertificateFactory factory = CertificateFactory.getI
nstance("X509");
            X509Certificate x509 = (X509Certificate)factory.gene
rateCertificate(is);
            String alias = x509.getSubjectDN().getName();
            ks.setCertificateEntry(alias, x509);
        } finally {
            try { is.close(); } catch (IOException e) { /* This
is sample, so omit the exception process
            */ }
        }
    }
}
```

PrivateCertificateHttpsActivity.java

```
package org.jssec.android.https.privatecertificate;

import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;

    public class PrivateCertificateHttpsActivity extends Activity
 {

    private EditText mUrlBox;
    private TextView mMsgBox;
    private ImageView mImgBox;
    private AsyncTask<String, Void, Object> mAsyncTask ;

    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mUrlBox = (EditText)findViewById(R.id.urlbox);
        mMsgBox = (TextView)findViewById(R.id.msgbox);
        mImgBox = (ImageView)findViewById(R.id.imageview);
    }

    @Override
    protected void onPause() {
        // After this, Activity may be discarded, so cancel asyn
chronous process in advance.
        if (mAsyncTask != null) mAsyncTask.cancel(true);
        super.onPause();
    }

    public void onClick(View view) {
        String url = mUrlBox.getText().toString();
        mMsgBox.setText(url);
        mImgBox.setImageBitmap(null);
        // Cancel, since the last asynchronous process might hav
e not been finished yet.
        if (mAsyncTask != null) mAsyncTask.cancel(true);
        // Since cannot communicate through UI thread, communica
te by worker thread by AsynchTask.
        mAsyncTask = new PrivateCertificateHttpsGet(this) {
            @Override
            protected void onPostExecute(Object result) {
                // Process the communication result through UI t
hread.
                if (result instanceof Exception) {
                    Exception e = (Exception)result;
                    mMsgBox.append("¥nException occurs¥n" + e.to
String());
                } else {
                    byte[] data = (byte[])result;
                    Bitmap bmp = BitmapFactory.decodeByteArray(d
ata, 0, data.length);
                    mImgBox.setImageBitmap(bmp);
                }
            }
        }.execute(url);
        // Pass URL and start asynchronization process
    }
}
```

5.4.2 规则书

使用 HTTP/S 通信时，遵循以下规则：

5.4.2.1 必须通过 HTTPS 通信发送/接收敏感信息（必需）

在 HTTP 事务中，发送和接收的信息可能被嗅探或篡改，并且连接的服务器可能被伪装。 敏感信息必须通过 HTTPS 通信发送/接收。

5.4.2.2 必须小心和安全地处理通过 HTTP 接收到的数据（必需）

HTTP 通信中收到的数据可能由攻击者利用应用的漏洞产生。 因此，你必须假定应用收到任何值和格式的数据，然后小心实现数据处理来处理收到的数据，以免造成任何漏洞。此外，你不应该盲目信任来自 HTTPS 服务器的数据。 由于 HTTPS 服务器可能由攻击者制作，或者收到的数据可能在 HTTPS 服务器的其他位置制作。 请参阅“3.2 小心和安全地处理输入数据”。

5.4.2.3 `SSLException` 必须适当处理，例如通知用户（必需）

在 HTTPS 通信中，当服务器证书无效或通信处于中间人攻击下时， `SSLException` 会作为验证错误产生。 所以你必须为 `SSLException` 实现适当的异常处理。 通知用户通信失败，记录故障等，可被认为是异常处理的典型实现。 另一方面，在某些情况下可能不需要特别通知用户。 因为如何处理 `SSLException` 取决于应用规范和特性，你需要首先考虑彻底后再确定它。

如上所述，当 `SSLException` 产生时，应用可能受到中间人的攻击，所以它不能实现为，试图通过例如 HTTP 的非安全协议再次发送/接收敏感信息。

5.4.2.4 不要创建自定义的 `TrustManager` （必需）

仅仅更改用于验证服务器证书的 `KeyStore` ，就足以通过 HTTPS ，与例如自签名证书的私有证书进行通信。 但是，正如在“5.4.3.3 禁用证书验证的危险代码”中所解释的那样，在因特网上有很多危险的 `TrustManager` 实现，与用于这种目的的示例代码一样。 通过引用这些示例代码而实现的应用可能有此漏洞。

当你需要通过 HTTPS 与私有证书进行通信时，请参阅“5.4.1.3 通过 HTTPS 与私有证书进行通信”中的安全示例代码。

当然，自定义的 `TrustManager` 可以安全地实现，但需要足够的加密处理和加密通信知识，以免执行存在漏洞的代码。 所以这个规则应为（必需）。

5.4.3 高级话题

5.4.3.1 如何创建私有证书并配置服务器

在本节中，将介绍如何在 Linux（如 Ubuntu 和 CentOS）中创建私有证书和配置服务器。私有证书是指私人签发的服务器证书，并由 Cybertrust 和 VeriSign 等可信第三方证书机构签发的服务器证书通知。

创建私有证书机构

首先，你需要创建一私有证书机构来颁发私有证书。私有证书机构是指私有创建的证书机构以及私有证书。你可以使用单个私有证书机构颁发多个私有证书。存储私有证书机构的个人电脑应严格限制为只能由可信的人访问。

为了创建私有证书机构，必须创建两个文件，例如以下 shell 脚本 `newca.sh` 和设置文件 `openssl.cnf` ，然后执行它们。在 shell 脚本中， `CASTART` 和 `CAEND` 代表证书机构的有效期， `CASUBJ` 代表证书机构的名称。所以这些值需要根据你创建的证书机构进行更改。在执行 shell 脚本时，访问证书机构的密码总共需要 3 次，所以你需要每次都输入它。

newca.sh -- 创建证书机构的 Shell 脚本

```
#!/bin/bash

umask 0077

CONFIG=openssl.cnf
CATOP=./CA
CAKEY=cakey.pem
CAREQ=careq.pem
CACERT=cacert.pem
CAX509=cacert.crt
CASTART=130101000000Z # 2013/01/01 00:00:00 GMT
CAEND=230101000000Z # 2023/01/01 00:00:00 GMT
CASUBJ="/CN=JSSEC Private CA/O=JSSEC/ST=Tokyo/C=JP"

mkdir -p ${CATOP}
mkdir -p ${CATOP}/certs
mkdir -p ${CATOP}/crl
mkdir -p ${CATOP}/newcerts
mkdir -p ${CATOP}/private
touch ${CATOP}/index.txt

openssl req -new -newkey rsa:2048 -sha256 -subj "${CASUBJ}" ¥
    -keyout ${CATOP}/private/${CAKEY} -out ${CATOP}/${CAREQ}
openssl ca -selfsign -md sha256 -create_serial -batch ¥
    -keyfile ${CATOP}/private/${CAKEY} ¥
    -startdate ${CASTART} -enddate ${CAEND} -extensions v3_ca ¥
    -in ${CATOP}/${CAREQ} -out ${CATOP}/${CACERT} ¥
    -config ${CONFIG}
openssl x509 -in ${CATOP}/${CACERT} -outform DER -out ${CATOP}/$
{CAX509}
```

openssl.cnf -- 2 个 shell 脚本共同参照的 openssl 命令的设置文件

```
[ ca ]
default_ca = CA_default # The default ca section

[ CA_default ]
dir = ./CA # Where everything is kept
certs = $dir/certs # Where the issued certs are kept
crl_dir = $dir/crl # Where the issued crl are kept
database = $dir/index.txt # database index file.
                    #Proprietary-defined _subject = no # Set to
'no' to allow creation of
                    # several ctificates with same subject.
new_certs_dir = $dir/newcerts # default place for new certs.
certificate = $dir/cacert.pem # The CA certificate
serial = $dir/serial # The current serial number
crlnumber = $dir/crlnumber # the current crl number
                    # must be commented out to leave a V1 CRL
crl = $dir/crl.pem # The current CRL
private_key = $dir/private/cakey.pem# The private key
RANDFILE = $dir/private/.rand # private random number file
x509_extensions = usr_cert # The extentions to add the cert
name_opt = ca_default # Subject Name options
cert_opt = ca_default # Certificate field options
policy = policy_match

[ policy_match ]
countryName = match
stateOrProvinceName = match
organizationName = supplied
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[ usr_cert ]
basicConstraints=CA:FALSE
nsComment = "OpenSSL Generated Certificate"
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer

[ v3_ca ]
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer
basicConstraints = CA:true
```

创建私有证书

为了创建私有证书，你必须创建一个 shell 脚本并执行它，像下面的 `newca.sh` 一样。在 shell 脚本中，`SVSTART` 和 `SVEND` 代表私有证书的有效期，`SVSUBJ` 代表 Web 服务器的名称，所以这些值需要根据目标 Web 服务器而更改。尤其是，你需要确保不要将错误的主机名设置为 `SVSUBJ` 的 `/CN` ，它指定了 Web 服务器

主机名。 在执行 shell 脚本时，会询问访问证书机构的密码，因此你需要输入你在创建私有证书机构时设置的密码。 之后， `y / n` 总共被询问 2 次，每次需要输入 `y` 。

newsv.sh -- 签发私有证书的 Shell 脚本

```
#!/bin/bash

umask 0077

CONFIG=openssl.cnf
CATOP=./CA
CAKEY=cakey.pem
CACERT=cacert.pem
SVKEY=svkey.pem
SVREQ=svreq.pem
SVCERT=svcert.pem
SVX509=svcert.crt
SVSTART=130101000000Z # 2013/01/01 00:00:00 GMT
SVEND=230101000000Z # 2023/01/01 00:00:00 GMT
SVSUBJ="/CN=selfsigned.jssec.org/O=JSSEC Secure Cofing Group/ST=
Tokyo/C=JP"

openssl genrsa -out ${SVKEY} 2048
openssl req -new -key ${SVKEY} -subj "${SVSUBJ}" -out ${SVREQ}
openssl ca -md sha256 ¥
    -keyfile ${CATOP}/private/${CAKEY} -cert ${CATOP}/${CACERT}
¥
    -startdate ${SVSTART} -enddate ${SVEND} ¥
    -in ${SVREQ} -out ${SVCERT} -config ${CONFIG}
openssl x509 -in ${SVCERT} -outform DER -out ${SVX509}
```

执行上面的 shell 脚本后，Web 服务器的 `svkey.pem` （私钥文件）和 `svcert.pem` （私有证书文件）都在工作目录下生成。 当 Web 服务器是 Apache 时，你将在配置文件中指定 `prikey.pem` 和 `cert.pem` ，如下所示。

```
SSLCertificateFile "/path/to/svcert.pem"
SSLCertificateKeyFile "/path/to/svkey.pem"
```

5.4.3.2 将私有证书机构的根证书安装到 Android 操作系统的证书商店

在示例代码“5.4.1.3 通过使用私有证书的 HTTPS 进行通信”中，介绍了通过将根证书安装到应用中，使用私有证书建立应用到 Web 服务器的 HTTPS 会话的方法。 本节将介绍通过将根证书安装到 Android OS 中，建立使用私有证书的所有应用到 Web 服务器的 HTTPS 会话的方法。 请注意，你安装的所有东西，应该是由可信证书机构颁发的证书，包括你自己的证书机构。

首先，你需要将根证书文件 `cacert.crt` 复制到 Android 设备的内部存储器中。你也可以从 https://selfsigned.jssec.org/cacert.crt 获取示例代码中使用的根证书文件。

然后，你将从 Android 设置中打开安全页面，然后你可以按如下方式在 Android 设备上安装根证书。

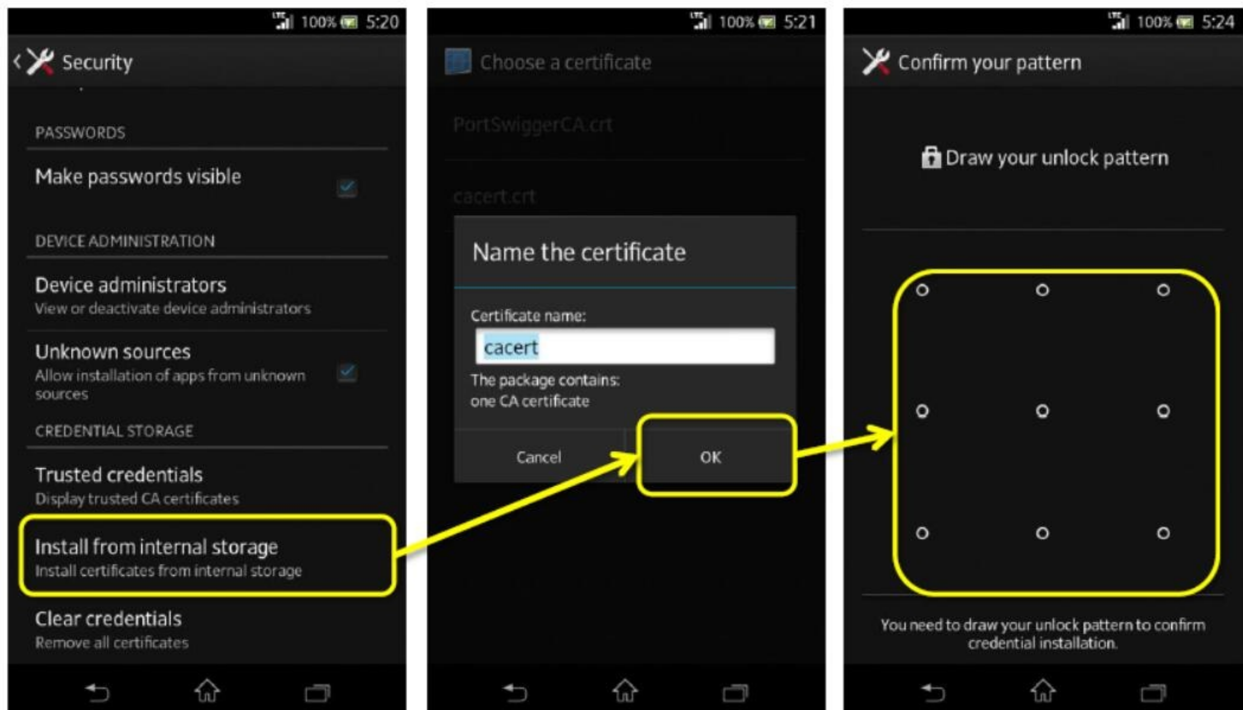


Figure 5.4-2 Steps to install root certificate of private certificate authority

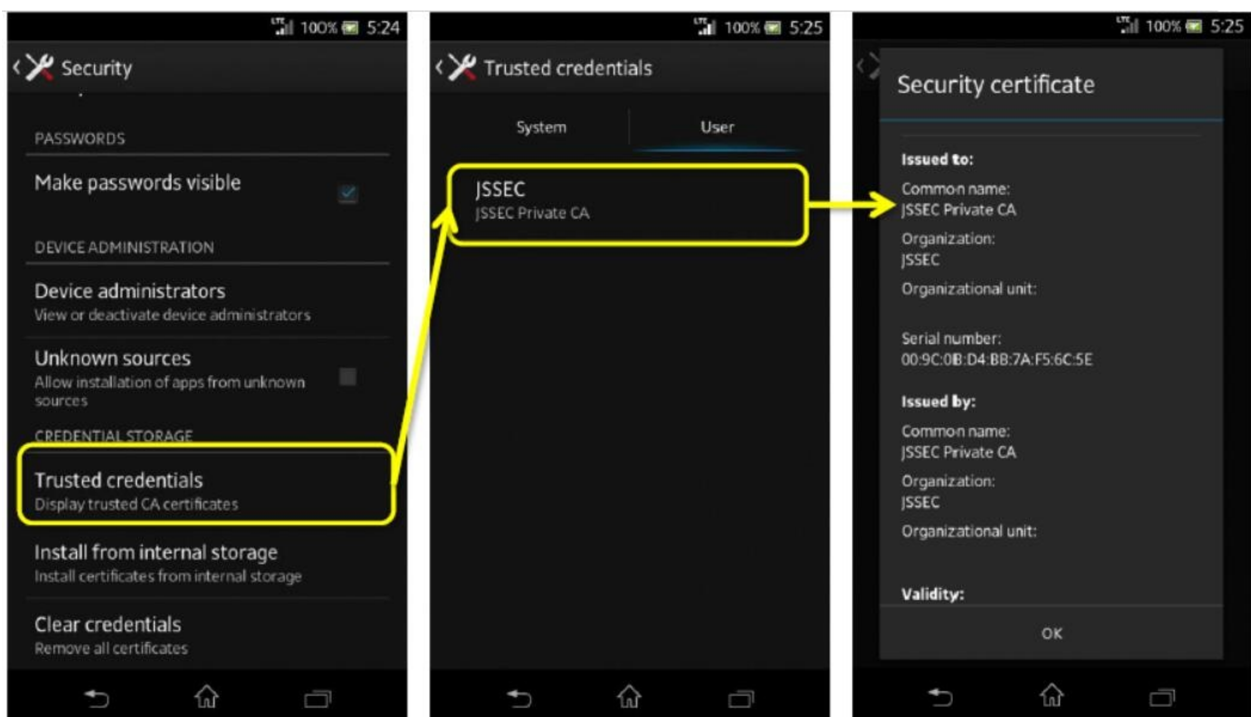


Figure 5.4-3 Checking if root certificate is installed or not

在 Android 操作系统中安装根证书后，所有应用都可以正确验证证书机构颁发的每个私有证书。 下图显示了在 Chrome 浏览器中显示 https://selfsigned.jssec.org/droid_knight.png 时的示例。

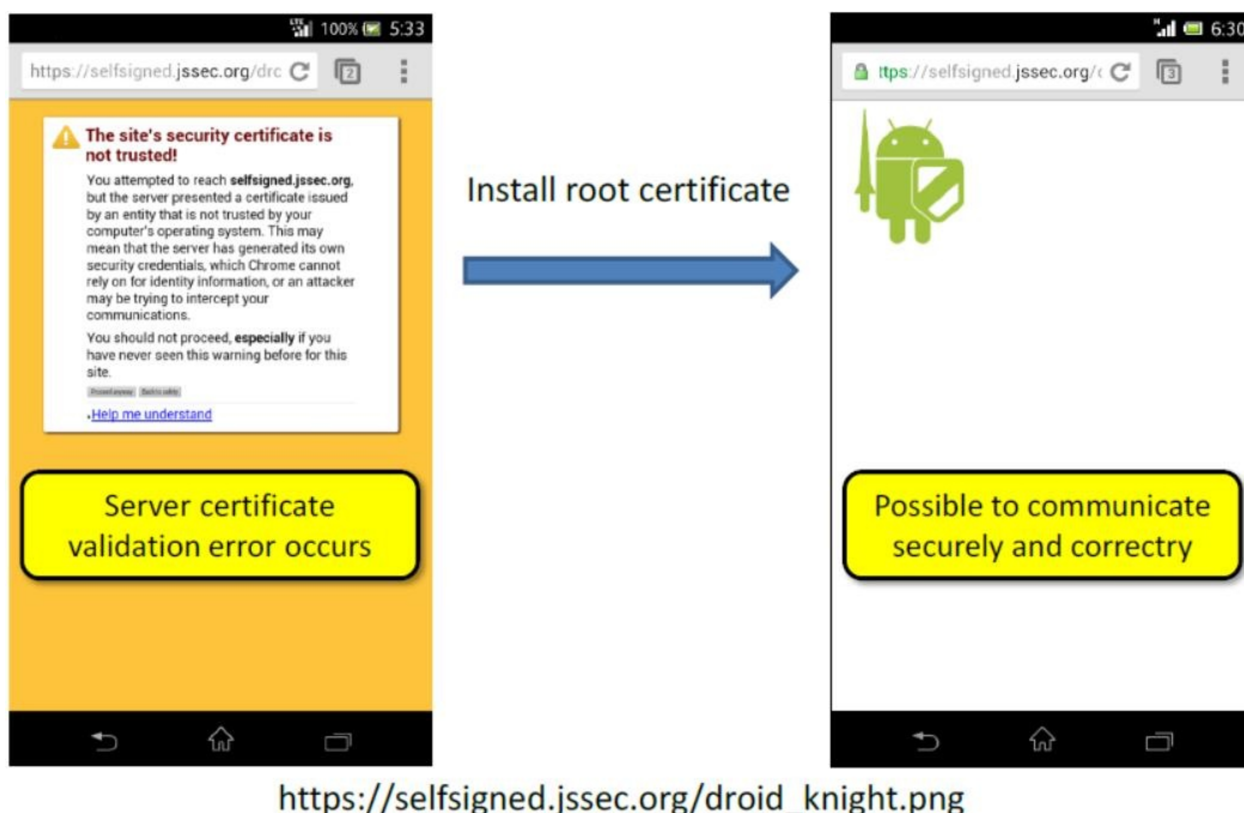


Figure 5.4-4 Once root certificate installed, private certificates can be verified correctly.

通过以这种方式安装根证书，即使是使用示例代码“5.4.1.2 通过 HTTPS 通信”的应用，也可以通过 HTTPS 正确连接到使用私有证书操作的 Web 服务器。

5.4.3.3 禁止证书验证的危险代码

互联网上发现了很多不正确的示例（代码片段），它们允许应用在证书验证错误发生后，通过 HTTPS 与 Web 服务器继续通信。 由于它们作为一种方式而引入，通过 HTTPS 与使用私有证书的 Web 服务器进行通信，因此开发人员通过复制和粘贴使用这些示例代码，创建了许多应用。 不幸的是，他们中的大多数容易受到中间人攻击。 正如本文前面所述，“2012 年，Android 应用中 HTTPS 通信实现中的许多缺陷被指出”，许多 Android 应用已经实现了这种易受攻击的代码。

下面显示了 HTTPS 通信的几个存在漏洞的代码片段。 当你找到此类代码片段时，强烈建议替换为“5.4.1.3 通过 HTTPS 与私有证书进行通信”的示例代码。

风险：创建空 `TrustManager` 时的情况

```
TrustManager tm = new X509TrustManager() {
    @Override
    public void checkClientTrusted(X509Certificate[] chain,
        String authType) throws CertificateException {
        // Do nothing -> accept any certificates
    }

    @Override
    public void checkServerTrusted(X509Certificate[] chain,
        String authType) throws CertificateException {
        // Do nothing -> accept any certificates
    }

    @Override
    public X509Certificate[] getAcceptedIssuers() {
        return null;
    }
};
```

风险：创建空 `HostnameVerifier` 时的情况

```
HostnameVerifier hv = new HostnameVerifier() {
    @Override
    public boolean verify(String hostname, SSLSession session) {
        // Always return true -> Accespt any host names
        return true;
    }
};
```

风险：使用 `ALLOW_ALL_HOSTNAME_VERIFIER` 的情况

```
SSLSocketFactory sf;

[...]

sf.setHostnameVerifier(SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIF
IER);
```

5.4.3.4 HTTP 请求头配置的注意事项

如果你希望为 HTTP 或 HTTPS 通信指定你自己的单个 HTTP 请求头，请使用 `URLConnection` 类中的 `setRequestProperty()` 或 `addRequestProperty()` 方法。如果你使用从外部来源接收的输入数据作为这些方法的参数，则必须实施 HTTP 协议头注入保护。HTTP 协议头注入攻击的第一步，是在输入数据中包含回车代码（在 HTTP 头中用作分隔符）。因此，必须从输入数据中删除所有回车代码。

配置 HTTP 请求头

```
public byte[] openConnection(String strUrl, String strLanguage,
String strCookie) {
    // HttpURLConnection is a class derived from URLConnection
    HttpURLConnection connection;
    try {
        URL url = new URL(strUrl);
        connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod("GET");
        // *** POINT *** When using input values in HTTP request
 headers,
        // check the input data in accordance with the applicati
on's requirements
        // (see Section 3.2: Handling Input Data Carefully and S
ecurely)
        if (strLanguage.matches("^[a-zA-Z ,-]+$")) {
            connection.addRequestProperty("Accept-Language", str
Language);
        } else {
            throw new IllegalArgumentException("Invalid Language
 : " + strLanguage);
        }
        // *** POINT *** Or URL-encode the input data (as approp
riate for the purposes of the app in
        queestion)
        connection.setRequestProperty("Cookie", URLEncoder.encod
e(strCookie, "UTF-8"));
        connection.connect();

        [...]
```

5.4.3.5 用于固定的注解和实现示例

当应用使用 HTTPS 通信时，在通信开始时执行的握手过程中的一个步骤是，检查从远程服务器发送的证书是否由第三方证书机构签署。但是，攻击者可能会从第三方认证代理获取不合适的证书，或者可能从证书机构获取签署的密钥来构造不合适的证书。在这种情况下，应用将无法在握手过程中检测到攻击，即使在攻击者建立不正确的服务器或中间人攻击的情况下也是如此 - 因此， ，可能会造成损失。

固定技术是一种有效的策略，可以防止不正当的第三方证书机构使用这些类型的证书，来进行中间人攻击。在这种方法中，远程服务器的证书和公钥被预先存储在一个应用中，并且这个信息用于握手过程，以及握手过程完成后的重新测试。

如果第三方证书机构（公钥基础设施的基础）的可信度受到损害，则可以使用固定来恢复通信的安全性。 应用开发人员应评估自己的应用处理的资产级别，并决定是否实现这些测试。

在握手过程中使用存储在应用中的证书和公钥

为了在握手过程中，使用存储在应用中的远程服务器证书或公钥中包含的信息，应用必须创建包含此信息的，自己的 `KeyStore` 并在通信时使用它。 如上所述，即使在使用来自不正当的第三方证书机构的证书的，中间人攻击的情况下，这也将允许应用检测握手过程中的不当行为。 请参阅“5.4.1.3 使用 HTTPS 与私有证书进行通信”一节中介绍的示例代码，了解建立应用自己的 `KeyStore` 来执行 HTTPS 通信的详细方法。

握手过程完成后，使用应用中存储的证书和公钥信息进行重新测试

为了在握手过程完成后重新测试远程服务器，应用首先会获得证书链，它在握手过程中受到系统测试和信任，然后比较该证书链和预先存储在应用中的信息。 如果比较结果表明它与应用中存储的信息一致，则可以允许通信进行；否则，应该中止通信过程。 但是，如果应用使用下面列出的方法，尝试获取在握手期间受系统信任的证书链，则应用可能无法获得预期的证书链，从而存在固定可能无法正常工作的风险 [26]。

> [26] 这篇文章详细解释了风险：https://www.cigital.com/blog/ineffective-certificate-pinning-implementations/。

- `javax.net.ssl.SSLSession.getPeerCertificates()`
- `javax.net.ssl.SSLSession.getPeerCertificateChain()`

这些方法返回的东西，不是在握手过程中受系统信任的证书链，而是应用从通信伙伴本身接收到的证书链。 因此，即使中间人攻击导致证书链中附加不正当证书机构的证书，上述方法也不会返回握手期间受系统信任的证书; 相反，应用最初试图连接的服务器的证书也将同时返回。 由于固定，这个证书将等同于预先存储在应用中的证书；因此重新测试它不会检测到任何不当行为。 由于这个以及其他类似的原因，在握手后执行重新测试时，最好避免使用上述方法。

在 Android 版本4.2（API 级别 17）及更高版本中，使用 `net.http.X509TrustManagerExtensions中的checkServerTrusted()` 方法，将允许应用仅获取握手期间受系统信任的证书链。

一个示例，展示了使用 `X509TrustManagerExtensions` 的固定

```
// Store the SHA-256 hash value of the public key included in th
e correct certificate for the remote server (pinning)
private static final Set<String> PINS = new HashSet<>(Arrays.asL
ist(
    new String[] {
        "d9b1a68fceaa460ac492fb8452ce13bd8c78c6013f989b76f186b1c
bba1315c1",
        "cd13bb83c426551c67fabcff38d4496e094d50a20c7c15e886c151d
eb8531cdc"
    }
));

// Communicate using AsyncTask work threads
protected Object doInBackground(String... strings) {

    [...]
```

```
    // Obtain the certificate chain that was trusted by the syst
em by testing during the handshake
    X509Certificate[] chain = (X509Certificate[]) connection.get
ServerCertificates();
    X509TrustManagerExtensions trustManagerExt = new X509TrustMa
nagerExtensions((X509TrustManager) (trus
    tManagerFactory.getTrustManagers()[0]));
    List<X509Certificate> trustedChain = trustManagerExt.checkSe
rverTrusted(chain, "RSA", url.getHost());
    // Use public-key pinning to test
    boolean isValidChain = false;
    for (X509Certificate cert : trustedChain) {
        PublicKey key = cert.getPublicKey();
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        String keyHash = bytesToHex(md.digest(key.getEncoded()))
;
        // Compare to the hash value stored by pinning
        if(PINS.contains(keyHash)) isValidChain = true;
    }
    if (isValidChain) {
        // Proceed with operation
    } else {
        // Do not proceed with operation
    }

    [...]
}

private String bytesToHex(byte[] bytes) {
    StringBuilder sb = new StringBuilder();
    for (byte b : bytes) {
        String s = String.format("%02x", b);
        sb.append(s);
    }
    return sb.toString();
}
```

5.4.3.6 使用 Google Play 服务解决 OpenSSL 漏洞的策略

Google Play 服务（版本 5.0 和更高）提供了一个称为 Provider Installer 的框架。这可以用于解决安全供应器中的漏洞，它是 OpenSSL 和其他加密相关技术的实现。 详细信息请参见 “5.6.3.5 通过 Google Play 服务解决安全供应器的漏洞”。

5.4.3.7 网络安全配置

Android 7.0（API Level 24）引入了一个称为“网络安全配置”的框架，允许各个应用为网络通信配置它们自己的安全设置。 通过使用此框架，应用可以轻松集成各种技术，来提高应用安全性，不仅包括与私钥证书和公钥固定的 HTTPS 通信，还可防止未加密（HTTP）通信，以及仅在调试过程中启用的私钥证书 [27]。

> [27] 网络安全配置的更多信息，请见 https://developer.android.com/training/articles/security-config.html 。

只需通过配置 `xml` 文件中的设置，即可访问网络安全配置提供的各种功能，它们可应用于整个应用的 HTTP 和 HTTPS 通信。 这消除了修改应用代码或执行任何额外操作的需要，简化了实现并提供了防范组合错误或漏洞的有效方法。

使用私有证书通过 HTTPS 进行通信

“5.4.1.3 通过 HTTPS 与有证书进行通信”部分介绍了与私有证书（例如自签名证书或公司内部证书）的 HTTPS 通信的示例代码。 但是，通过使用网络安全配置，开发人员可以在“5.4.1.2 通过 HTTPS 进行通信”的示例代码中使用私有证书，而无需实现。

使用私有证书与特定域进行通信

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config>
        <domain includeSubdomains="true">jssec.org</domain>
        <trust-anchors>
            <certificates src="@raw/private_ca" />
        </trust-anchors>
    </domain-config>
</network-security-config>
```

在上面的示例中，用于通信的私有证书（ `private_ca` ）可以作为资源存储在应用中，带有使用条件及其在 `.xml` 文件中描述的适用范围。 通过使用 `<domain-config>` 标签，私有证书可以仅仅应用于特定域。 为了对应用执行的所有 HTTPS 通信使用私有证书，请使用 `<base-config>` 标签，如下所示。

对应用执行的所有 HTTPS 通信使用私人证书

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config>
        <trust-anchors>
            <certificates src="@raw/private_ca" />
        </trust-anchors>
    </base-config>
</network-security-config>
```

固定

我们在“5.4.3.5 针对固定的注解和实现示例”中提到了公钥固定。通过使用网络安全配置，如下例所示，你不必在代码中实现认证过程; 相反， `xml` 文件中的规范足以确保正确的认证。

对 HTTPS 通信使用公钥固定

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config>
    <domain includeSubdomains="true">jssec.org</domain>
        <pin-set expiration="2018-12-31">
            <pin digest="SHA-256">e30Lky+iWK21yHSls5DJoRzNikOdvQ
UOGXvurPidc2E=</pin>
            <!-- 用于备份 -->
            <pin digest="SHA-256">fwza0LRMXouZHRC8Ei+4PyuldPDcf3
UKgO/04cDM1oE=</pin>
        </pin-set>
    </domain-config>
</network-security-config>
```

上面 `<pin>` 标签描述的数量，是用于固定的公钥的 base64 编码哈希值。 唯一支持的散列函数是 SHA-256。

防止未加密（HTTP）通信

使用网络安全配置可以阻止应用进行 HTTP 通信（未加密通信）。

防止未加密（HTTP）通信

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="false">
    <domain includeSubdomains="true">jssec.org</domain>
    </domain-config>
</network-security-config>
```

在上面的例子中，我们在 `<domain-config>` 标签中指定了属性 `cleartextTrafficPermitted="false"` 。 这可以防止指定域的 HTTP 通信，从而强制使用 HTTPS 通信。 在 `<base-config>` 标记中包含此属性设置，将会阻止所有域的 HTTP 通信 [28]。但请谨慎注意，此设置不适用于 `WebView` 。

> [28] 网络安全配置如何为非 HTTP 连接工作，请参阅以下 API 参考。

特地用于调试目的的私有证书

为了在应用开发过程中进行调试，开发人员可能希望使用私有证书，与某些 HTTPS 服务器进行通信，它们由于应用开发目的而存在。 在这种情况下，开发人员必须注意确保没有危险的实现（包括禁用证书认证的代码）被合并到应用中；这在“5.4.3.3 禁用证书验证的危险代码”一节中讨论。 在网络安全配置中，可以按照下面的示例来配置，来规定一组仅在调试时才使用的证书（仅当 `AndroidManifest.xml` 文件中的 `android:debuggable` 设置为 `true` 时）。 这消除了危险代码可能无意中保留在应用的发行版中的风险，因此是防止漏洞的有效手段。

仅在调试时使用私有证书

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <debug-overrides>
        <trust-anchors>
            <certificates src="@raw/private_cas" />
        </trust-anchors>
    </debug-overrides>
</network-security-config>
```

5.5 处理隐私数据

近年来，“隐私设计”概念已被提出，作为保护隐私数据的全球趋势。基于这一概念，各国政府正在推动隐私保护立法。

在智能手机中使用用户数据的应用必须采取措施，来确保用户可以安全地使用应用，而不必担心隐私和个人数据。这些步骤包括适当处理用户数据，并要求用户选择应用是否可以使用某些数据。为此，每个应用必须准备并显示应用隐私策略，指明应用将使用哪些信息，以及如何使用该信息；而且，在获取和使用某些信息时，应用必须首先向用户请求许可。请注意，应用隐私政策与过去可能存在的其他文档（例如“个人数据保护政策”或“使用条款”）不同，且必须与任何此类文档分开创建。

创建和执行隐私政策的详细信息，请参见日本总务省（MIC）发布的文档《Smartphone Privacy Initiative》和《Smartphone Privacy Initiative II》（JMIC 的 SPI）。

本节中使用的术语在“5.5.3.2 术语表”中以文本定义。

5.5.1 示例代码

在准备应用的隐私政策时，你可以使用“协助创建应用隐私政策的工具” [29]。 这些工具以 HTML 格式和 XML 格式输出两个文件 - 应用隐私策略的摘要版本和详细版本。 这些文件的 HTML 和 XML 内容符合 MIC SPI 的建议，包括搜索标签等特性。在下面的示例代码中，我们将演示此工具的用法，并使用由这个工具产生的 HTML 文件来展示程序隐私策略。

> [29] http://www.kddilabs.jp/tech/public-tech/appgen.html

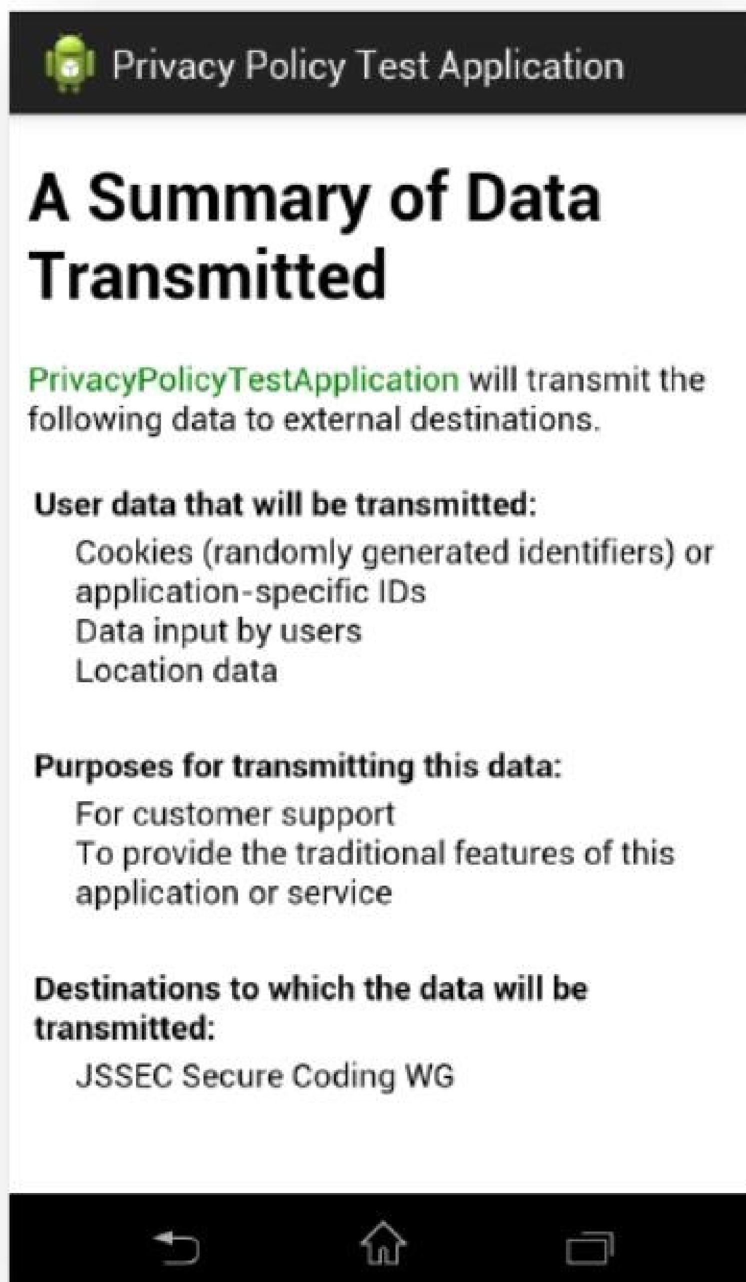


Figure 5.5-1 Sample of Abstract Application Privacy Policy

更具体地说，你可以使用以下流程图来确定使用哪个示例代码。

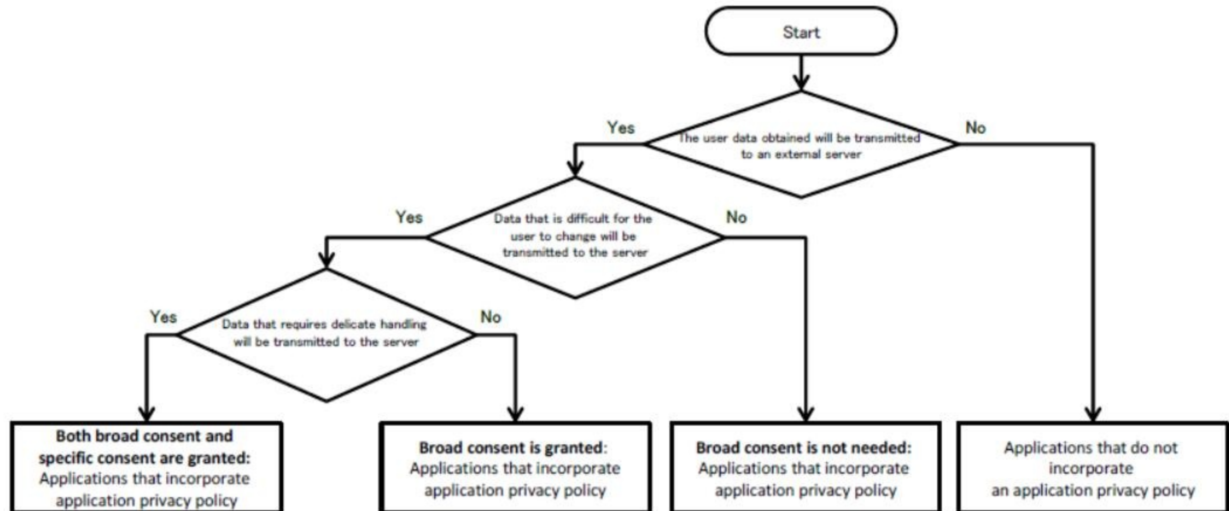


Figure 5.5-2 Flow Figure to select sample code of handling privacy data

这里，“广泛同意”一词，指代广泛许可，由用户在应用的首次加载时，通过展示和查看程序隐私策略授予应用，用于应用将用户数据传输到服务器。 相反，短语“特定同意”指代在传输特定用户数据之前，立即获得的预先同意。

5.5.1.1 授予广泛同意和特定同意：包含应用隐私政策的应用

要点：

1. 首次加载（或应用更新）时，获得广泛同意，来传输将由应用处理的用户数据。
2. 如果用户未授予广泛同意，请勿传输用户数据。
3. 在传输需要特别细致的处理的用户数据之前获得特定同意。
4. 如果用户未授予特定同意，请勿传输相应的数据。
5. 向用户提供可以查看应用隐私策略的方法。
6. 提供通过用户操作删除传输的数据的方法。
7. 提供通过用户操作停止数据传输的方法。
8. 使用 UUID 或 cookie 来跟踪用户数据。
9. 将应用隐私策略的摘要版本放置在素材文件夹中。

MainActivity.java

```
package org.jssec.android.privacypolicy;
import java.io.IOException;
import org.json.JSONException;
import org.json.JSONObject;
import org.jssec.android.privacypolicy.ConfirmFragment.DialogLis
tener;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GooglePlayServicesClient;
import com.google.android.gms.common.GooglePlayServicesUtil;
import com.google.android.gms.location.LocationClient;
import android.location.Location;
```

```
import android.os.AsyncTask;
import android.os.Bundle;
import android.content.Intent;
import android.content.IntentSender;
import android.content.SharedPreferences;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;
public class MainActivity extends FragmentActivity
    implements GooglePlayServicesClient.ConnectionCallbacks,
    GooglePlayServicesClient.OnConnectionFailedListener, DialogL
istener {

    private static final String BASE_URL = "https://www.example.
com/pp";
    private static final String GET_ID_URI = BASE_URL + "/get_id
.php";
    private static final String SEND_DATA_URI = BASE_URL + "/sen
d_data.php";
    private static final String DEL_ID_URI = BASE_URL + "/del_id
.php";
    private static final String ID_KEY = "id";
    private static final String LOCATION_KEY = "location";
    private static final String NICK_NAME_KEY = "nickname";
    private static final String PRIVACY_POLICY_COMPREHENSIVE_AGR
EED_KEY = "privacyPolicyComprehensiveAgreed";
    private static final String PRIVACY_POLICY_DISCRETE_TYPE1_AG
REED_KEY = "privacyPolicyDiscreteType1Agreed";
    private static final String PRIVACY_POLICY_PREF_NAME = "priv
acypolicy_preference";
    private static final int CONNECTION_FAILURE_RESOLUTION_REQUE
ST = 257;
    private String UserId = "";
    private LocationClient mLocationClient = null;
    private final int DIALOG_TYPE_COMPREHENSIVE_AGREEMENT = 1;
    private final int DIALOG_TYPE_PRE_CONFIRMATION = 2;
    private static final int VERSION_TO_SHOW_COMPREHENSIVE_AGREE
MENT_ANEW = 1;

    private TextWatcher watchHandler = new TextWatcher() {

        @Override
        public void beforeTextChanged(CharSequence s, int start,
 int count, int after) {
```

```
        }

        @Override
        public void onTextChanged(CharSequence s, int start, int
 before, int count) {
            boolean buttonEnable = (s.length() > 0);
            MainActivity.this.findViewById(R.id.buttonStart).set
Enabled(buttonEnable);
        }

        @Override
        public void afterTextChanged(Editable s) {
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Fetch user ID from serverFetch user ID from server
        new GetDataAsyncTask().execute();
        findViewById(R.id.buttonStart).setEnabled(false);
        ((TextView) findViewById(R.id.editTextNickname)).addText
ChangedListener(watchHandler);
        int resultCode = GooglePlayServicesUtil.isGooglePlayServ
icesAvailable(this);
        if (resultCode == ConnectionResult.SUCCESS) {
            mLocationClient = new LocationClient(this, this, thi
s);
        }
    }

    @Override
    protected void onStart() {
        super.onStart();
        SharedPreferences pref = getSharedPreferences(PRIVACY_PO
LICY_PREF_NAME, MODE_PRIVATE);
        int privacyPolicyAgreed = pref.getInt(PRIVACY_POLICY_COM
PREHENSIVE_AGREED_KEY, -1);
        if (privacyPolicyAgreed <= VERSION_TO_SHOW_COMPREHENSIVE
_AGREEMENT_ANEW) {
            // *** POINT 1 *** On first launch (or application u
pdate), obtain broad consent to transmit user data that will be
handled by the application.
            // When the application is updated, it is only neces
sary to renew the user's grant of broad c
            onsent if the updated application will handle new ty
pes of user data.
            ConfirmFragment dialog = ConfirmFragment.newInstance
(R.string.privacyPolicy, R.string.agreeP
            rivacyPolicy, DIALOG_TYPE_COMPREHENSIVE_AGREEMENT);
            dialog.setDialogListener(this);
            FragmentManager fragmentManager = getSupportFragment
```

```
Manager();
                dialog.show(fragmentManager, "dialog");
            }
            // Used to obtain location data
            if (mLocationClient != null) {
                mLocationClient.connect();
            }
        }

        @Override
        protected void onStop() {
            if (mLocationClient != null) {
                mLocationClient.disconnect();
            }
            super.onStop();
        }

        public void onSendToServer(View view) {
            // Check the status of user consent.
            // Actually, it is necessary to obtain consent for each
user data type.
            SharedPreferences pref = getSharedPreferences(PRIVACY_PO
LICY_PREF_NAME, MODE_PRIVATE);
            int privacyPolicyAgreed = pref.getInt(PRIVACY_POLICY_DIS
CRETE_TYPE1_AGREED_KEY, -1);
            if (privacyPolicyAgreed <= VERSION_TO_SHOW_COMPREHENSIVE
_AGREEMENT_ANEW) {
                // *** POINT 3 *** Obtain specific consent before tr
ansmitting user data that requires particularly delicate handlin
g.
                ConfirmFragment dialog = ConfirmFragment.newInstance
(R.string.sendLocation, R.string.cofirmS
                endLocation, DIALOG_TYPE_PRE_CONFIRMATION);
                dialog.setDialogListener(this);
                FragmentManager fragmentManager = getSupportFragment
Manager();
                dialog.show(fragmentManager, "dialog");
            } else {
                // Start transmission, since it has the user consent
.
                onPositiveButtonClick(DIALOG_TYPE_PRE_CONFIRMATION);
            }
        }

        public void onPositiveButtonClick(int type) {
            if (type == DIALOG_TYPE_COMPREHENSIVE_AGREEMENT) {
                // *** POINT 1 *** On first launch (or application u
pdate), obtain broad consent to transmit user data that will be
handled by the application.
                SharedPreferences.Editor pref = getSharedPreferences
(PRIVACY_POLICY_PREF_NAME, MODE_PRIVATE).edit();
                pref.putInt(PRIVACY_POLICY_COMPREHENSIVE_AGREED_KEY,
 getVersionCode());
```

```
            pref.apply();
        } else if (type == DIALOG_TYPE_PRE_CONFIRMATION) {
            // *** POINT 3 *** Obtain specific consent before tr
ansmitting user data that requires particularly delicate handlin
g.
            if (mLocationClient != null && mLocationClient.isCon
nected()) {
                Location currentLocation = mLocationClient.getLa
stLocation();
                if (currentLocation != null) {
                    String locationData = "Latitude:" + currentL
ocation.getLatitude() + ", Longitude:" +
                        currentLocation.getLongitude();
                    String nickname = ((TextView) findViewById(R
.id.editTextNickname)).getText().toString();
                    Toast.makeText(MainActivity.this, this.getCl
ass().getSimpleName() + "¥n - nickname :
                        " + nickname + "¥n - location : " + loca
tionData, Toast.LENGTH_SHORT).show();
                    new SendDataAsyncTack().execute(SEND_DATA_UR
I, UserId, locationData, nickname);
                }
            }
            // Store the status of user consent.
            // Actually, it is necessary to obtain consent for e
ach user data type.
            SharedPreferences.Editor pref = getSharedPreferences
(PRIVACY_POLICY_PREF_NAME, MODE_PRIVATE).edit();
            pref.putInt(PRIVACY_POLICY_DISCRETE_TYPE1_AGREED_KEY
, getVersionCode());
            pref.apply();
        }
    }

    public void onNegativeButtonClick(int type) {
        if (type == DIALOG_TYPE_COMPREHENSIVE_AGREEMENT) {
            // *** POINT 2 *** If the user does not grant genera
l consent, do not transmit user data.
            // In this sample application we terminate the appli
cation in this case.
            finish();
        } else if (type == DIALOG_TYPE_PRE_CONFIRMATION) {
            // *** POINT 4 *** If the user does not grant specif
ic consent, do not transmit the correspon
            ding data.
            // The user did not grant consent, so we do nothing.
        }
    }

    private int getVersionCode() {
        int versionCode = -1;
        PackageManager packageManager = this.getPackageManager()
;
```

```
        try {
            PackageInfo packageInfo = packageManager.getPackageI
nfo(this.getPackageName(), PackageManager.GET_ACTIVITIES);
            versionCode = packageInfo.versionCode;
        } catch (NameNotFoundException e) {
            // This is sample, so omit the exception process
        }
        return versionCode;
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_show_pp:
                // *** POINT 5 *** Provide methods by which the
user can review the application privacy policy.
                Intent intent = new Intent();
                intent.setClass(this, WebViewAssetsActivity.clas
s);
                startActivity(intent);
                return true;
            case R.id.action_del_id:
                // *** POINT 6 *** Provide methods by which tran
smitted data can be deleted by user operations.
                new SendDataAsyncTack().execute(DEL_ID_URI, User
Id);
                return true;
            case R.id.action_donot_send_id:
                // *** POINT 7 *** Provide methods by which tran
smitting data can be stopped by user operations.
                // If the user stop sending data, user consent i
s deemed to have been revoked.
                SharedPreferences.Editor pref = getSharedPrefere
nces(PRIVACY_POLICY_PREF_NAME, MODE_PRIVATE).edit();
                pref.putInt(PRIVACY_POLICY_COMPREHENSIVE_AGREED_
KEY, 0);
                pref.apply();
                // In this sample application if the user data c
annot be sent by user operations,
                // finish the application because we do nothing.
                String message = getString(R.string.stopSendUser
Data);
                Toast.makeText(MainActivity.this, this.getClass(
).getSimpleName() + " - " + message, Toast.LENGTH_SHORT).show();
                finish();
                return true;
        }
```

```
        return false;
    }

    @Override
    public void onConnected(Bundle connectionHint) {
        if (mLocationClient != null && mLocationClient.isConnect
ed()) {
            Location currentLocation = mLocationClient.getLastLo
cation();
            if (currentLocation != null) {
                String locationData = "Latitude ¥t: " + currentL
ocation.getLatitude() + "¥n¥tLongitude ¥t: " + currentLocation.g
etLongitude();
                String text = "¥n" + getString(R.string.your_loc
ation_title) + "¥n¥t" + locationData;
                TextView appText = (TextView) findViewById(R.id.
appText);
                appText.setText(text);
            }
        }
    }

    @Override
    public void onConnectionFailed(ConnectionResult result) {
        if (result.hasResolution()) {
            try {
                result.startResolutionForResult(this, CONNECTION
_FAILURE_RESOLUTION_REQUEST);
            } catch (IntentSender.SendIntentException e) {
                e.printStackTrace();
            }
        }
    }

    @Override
    public void onDisconnected() {
        mLocationClient = null;
    }

    private class GetDataAsyncTask extends AsyncTask<String, Voi
d, String> {

        private String extMessage = "";

        @Override
        protected String doInBackground(String... params) {
            // *** POINT 8 *** Use UUIDs or cookies to keep trac
k of user data
            // In this sample we use an ID generated on the serv
er side
            SharedPreferences sp = getSharedPreferences(PRIVACY_
POLICY_PREF_NAME, MODE_PRIVATE);
            UserId = sp.getString(ID_KEY, null);
```

```
            if (UserId == null) {
                // No token in SharedPreferences; fetch ID from
server
                try {
                    UserId = NetworkUtil.getCookie(GET_ID_URI, "
", "id");
                } catch (IOException e) {
                    // Catch exceptions such as certification er
rors

                    extMessage = e.toString();
                }
                // Store the fetched ID in SharedPreferences
                sp.edit().putString(ID_KEY, UserId).commit();
            }
            return UserId;
        }

        @Override
        protected void onPostExecute(final String data) {
            String status = (data != null) ? "success" : "error"
;
            Toast.makeText(MainActivity.this, this.getClass().ge
tSimpleName() + " - " + status + " : " +
                extMessage, Toast.LENGTH_SHORT).show();
        }
    }

    private class SendDataAsyncTack extends AsyncTask<String, Vo
id, Boolean> {

        private String extMessage = "";

        @Override
        protected Boolean doInBackground(String... params) {
            String url = params[0];
            String id = params[1];
            String location = params.length > 2 ? params[2] : nu
ll;
            String nickname = params.length > 3 ? params[3] : nu
ll;
            Boolean result = false;
            try {
                JSONObject jsonData = new JSONObject();
                jsonData.put(ID_KEY, id);
                if (location != null)
                jsonData.put(LOCATION_KEY, location);
                if (nickname != null)
                jsonData.put(NICK_NAME_KEY, nickname);
                NetworkUtil.sendJSON(url, "", jsonData.toString(
));

                result = true;
            } catch (IOException e) {
                // Catch exceptions such as certification errors
```

```
                extMessage = e.toString();
            } catch (JSONException e) {
                extMessage = e.toString();
            }
            return result;
        }

        @Override
        protected void onPostExecute(Boolean result) {
            String status = result ? "Success" : "Error";
            Toast.makeText(MainActivity.this, this.getClass().ge
tSimpleName() + " - " + status + " : " +
                extMessage, Toast.LENGTH_SHORT).show();
        }
    }
}
```

ConfirmFragment.java

```
package org.jssec.android.privacypolicy;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.DialogFragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.TextView;

public class ConfirmFragment extends DialogFragment {
    private DialogListener mListener = null;

    public static interface DialogListener {
        public void onPositiveButtonClick(int type);
        public void onNegativeButtonClick(int type);
    }

    public static ConfirmFragment newInstance(int title, int sen
tence, int type) {
        ConfirmFragment fragment = new ConfirmFragment();
        Bundle args = new Bundle();
        args.putInt("title", title);
        args.putInt("sentence", sentence);
        args.putInt("type", type);
        fragment.setArguments(args);
        return fragment;
```

```
    }

    @Override
    public Dialog onCreateDialog(Bundle args) {
        // *** POINT 1 *** On first launch (or application updat
e), obtain broad consent to transmit user data that will be hand
led by the application.
        // *** POINT 3 *** Obtain specific consent before transm
itting user data that requires particularly delicate handling.
        final int title = getArguments().getInt("title");
        final int sentence = getArguments().getInt("sentence");
        final int type = getArguments().getInt("type");
        LayoutInflater inflater = (LayoutInflater) getActivity()
.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View content = inflater.inflate(R.layout.fragment_comfir
m, null);
        TextView linkPP = (TextView) content.findViewById(R.id.t
x_link_pp);
        linkPP.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // *** POINT 5 *** Provide methods by which the
user can review the application privacy policy.
                Intent intent = new Intent();
                intent.setClass(getActivity(), WebViewAssetsActi
vity.class);
                startActivity(intent);
            }
        });
        AlertDialog.Builder builder = new AlertDialog.Builder(ge
tActivity());
        builder.setIcon(R.drawable.ic_launcher);
        builder.setTitle(title);
        builder.setMessage(sentence);
        builder.setView(content);
        builder.setPositiveButton(R.string.buttonConsent, new Di
alogInterface.OnClickListener() {

            public void onClick(DialogInterface dialog, int whic
hButton) {
                if (mListener != null) {
                    mListener.onPositiveButtonClick(type);
                }
            }
        });
        builder.setNegativeButton(R.string.buttonDonotConsent, n
ew DialogInterface.OnClickListener() {

            public void onClick(DialogInterface dialog, int whic
hButton) {
                if (mListener != null) {
                    mListener.onNegativeButtonClick(type);
```

```
                        }
                    }
                });
                Dialog dialog = builder.create();
                dialog.setCanceledOnTouchOutside(false);
                return dialog;
        }

        @Override
        public void onAttach(Activity activity) {
                super.onAttach(activity);
                if (!(activity instanceof DialogListener)) {
                        throw new ClassCastException(activity.toString() + "
 must implement DialogListener.");
                }
                mListener = (DialogListener) activity;
        }

        public void setDialogListener(DialogListener listener) {
                mListener = listener;
        }
}
```

WebViewAssetsActivity.java

```
package org.jssec.android.privacypolicy;

import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;

public class WebViewAssetsActivity extends Activity {

    // *** POINT 9 *** Place a summary version of the applicatio
n privacy policy in the assets folder
    private static final String ABST_PP_URL = "file:///android_a
sset/PrivacyPolicy/app-policy-abst-privacypolicy-1.0.html";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_webview);
        WebView webView = (WebView) findViewById(R.id.webView);
        WebSettings webSettings = webView.getSettings();
        webSettings.setAllowFileAccess(false);
        webView.loadUrl(ABST_PP_URL);
    }
}
```

5.5.1.2 授予广泛同意：包含应用隐私政策的应用

要点：

1. 首次加载（或应用更新）时，获得广泛同意，来传输将由应用处理的用户数据。
2. 如果用户未授予广泛同意，请勿传输用户数据。
3. 向用户提供可以查看应用隐私策略的方法。
4. 提供通过用户操作删除传输的数据的方法。
5. 提供通过用户操作停止数据传输的方法。
6. 使用 UUID 或 cookie 来跟踪用户数据。
7. 将应用隐私策略的摘要版本放置在素材文件夹中。

MainActivity.java

```
package org.jssec.android.privacypolicynopreconfirm;

import java.io.IOException;
import org.json.JSONException;
import org.json.JSONObject;
import org.jssec.android.privacypolicynopreconfirm.MainActivity;
import org.jssec.android.privacypolicynopreconfirm.R;
import org.jssec.android.privacypolicynopreconfirm.ConfirmFragme
nt.DialogListener;
import android.os.AsyncTask;
import android.os.Bundle;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
import android.telephony.TelephonyManager;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends FragmentActivity implements Di
alogListener {

    private final String BASE_URL = "https://www.example.com/pp"
;
    private final String GET_ID_URI = BASE_URL + "/get_id.php";
    private final String SEND_DATA_URI = BASE_URL + "/send_data.
php";
    private final String DEL_ID_URI = BASE_URL + "/del_id.php";
```

```
    private final String ID_KEY = "id";
    private final String NICK_NAME_KEY = "nickname";
    private final String IMEI_KEY = "imei";
    private final String PRIVACY_POLICY_AGREED_KEY = "privacyPol
icyAgreed";
    private final String PRIVACY_POLICY_PREF_NAME = "privacypoli
cy_preference";
    private String UserId = "";
    private final int DIALOG_TYPE_COMPREHENSIVE_AGREEMENT = 1;
    private final int VERSION_TO_SHOW_COMPREHENSIVE_AGREEMENT_AN
EW = 1;

    private TextWatcher watchHandler = new TextWatcher() {

        @Override
        public void beforeTextChanged(CharSequence s, int start,
int count, int after) {
        }

        @Override
        public void onTextChanged(CharSequence s, int start, int
 before, int count) {
            boolean buttonEnable = (s.length() > 0);
            MainActivity.this.findViewById(R.id.buttonStart).set
Enabled(buttonEnable);
        }

        @Override
        public void afterTextChanged(Editable s) {
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Fetch user ID from serverFetch user ID from server
        new GetDataAsyncTask().execute();
        findViewById(R.id.buttonStart).setEnabled(false);
        ((TextView) findViewById(R.id.editTextNickname)).addText
ChangedListener(watchHandler);
    }

    @Override
    protected void onStart() {
        super.onStart();
        SharedPreferences pref = getSharedPreferences(PRIVACY_PO
LICY_PREF_NAME, MODE_PRIVATE);
        int privacyPolicyAgreed = pref.getInt(PRIVACY_POLICY_AGR
EED_KEY, -1);
        if (privacyPolicyAgreed <= VERSION_TO_SHOW_COMPREHENSIVE
_AGREEMENT_ANEW) {
            // *** POINT 1 *** On first launch (or application u
```

```
pdate), obtain broad consent to transmit user data that will be
handled by the application.
            // When the application is updated, it is only neces
sary to renew the user's grant of broad consent if the updated a
pplication will handle new types of user data.
            ConfirmFragment dialog = ConfirmFragment.newInstance
(R.string.privacyPolicy, R.string.agreePr
            ivacyPolicy, DIALOG_TYPE_COMPREHENSIVE_AGREEMENT);
            dialog.setDialogListener(this);
            FragmentManager fragmentManager = getSupportFragment
Manager();
            dialog.show(fragmentManager, "dialog");
        }
    }

    public void onSendToServer(View view) {
        String nickname = ((TextView) findViewById(R.id.editText
Nickname)).getText().toString();
        TelephonyManager tm = (TelephonyManager) getSystemServic
e(TELEPHONY_SERVICE);
        String imei = tm.getDeviceId();
        Toast.makeText(MainActivity.this, this.getClass().getSim
pleName() + "¥n - nickname : " + nickname + ", imei = " + imei,
Toast.LENGTH_SHORT).show();
        new SendDataAsyncTack().execute(SEND_DATA_URI, UserId, n
ickname, imei);
    }

    public void onPositiveButtonClick(int type) {
        if (type == DIALOG_TYPE_COMPREHENSIVE_AGREEMENT) {
            // *** POINT 1 *** On first launch (or application u
pdate), obtain broad consent to transmit
            user data that will be handled by the application.
            SharedPreferences.Editor pref = getSharedPreferences
(PRIVACY_POLICY_PREF_NAME, MODE_PRIVATE).edit();
            pref.putInt(PRIVACY_POLICY_AGREED_KEY, getVersionCod
e());
            pref.apply();
        }
    }

    public void onNegativeButtonClick(int type) {
        if (type == DIALOG_TYPE_COMPREHENSIVE_AGREEMENT) {
            // *** POINT 2 *** If the user does not grant genera
l consent, do not transmit user data.
            // In this sample application we terminate the appli
cation in this case.
            finish();
        }
    }

    private int getVersionCode() {
        int versionCode = -1;
```

```
        PackageManager packageManager = this.getPackageManager()
;
        try {
            PackageInfo packageInfo = packageManager.getPackageI
nfo(this.getPackageName(), PackageManager.GET_ACTIVITIES);
            versionCode = packageInfo.versionCode;
        } catch (NameNotFoundException e) {
            // This is sample, so omit the exception process
        }
        return versionCode;
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_show_pp:
                // *** POINT 3 *** Provide methods by which the
user can review the application privacy policy.
                Intent intent = new Intent();
                intent.setClass(this, WebViewAssetsActivity.clas
s);
                startActivity(intent);
                return true;
            case R.id.action_del_id:
                // *** POINT 4 *** Provide methods by which tran
smitted data can be deleted by user operation
                s.
                new SendDataAsyncTack().execute(DEL_ID_URI, User
Id);
                return true;
            case R.id.action_donot_send_id:
                // *** POINT 5 *** Provide methods by which tran
smitting data can be stopped by user operations.
                // If the user stop sending data, user consent i
s deemed to have been revoked.
                SharedPreferences.Editor pref = getSharedPrefere
nces(PRIVACY_POLICY_PREF_NAME, MODE_PRIVATE).edit();
                pref.putInt(PRIVACY_POLICY_AGREED_KEY, 0);
                pref.apply();
                // In this sample application if the user data c
annot be sent by user operations,
                // finish the application because we do nothing.
                String message = getString(R.string.stopSendUser
Data);
                Toast.makeText(MainActivity.this, this.getClass(
).getSimpleName() + " - " + message, Toast.L
                ENGTH_SHORT).show();
```

```
                finish();
                return true;
            }
            return false;
        }

        private class GetDataAsyncTask extends AsyncTask<String, Void
, String> {

            private String extMessage = "";

            @Override
            protected String doInBackground(String... params) {
                // *** POINT 6 *** Use UUIDs or cookies to keep trac
k of user data
                // In this sample we use an ID generated on the serv
er side
                SharedPreferences sp = getSharedPreferences(PRIVACY_
POLICY_PREF_NAME, MODE_PRIVATE);
                UserId = sp.getString(ID_KEY, null);
                if (UserId == null) {
                    // No token in SharedPreferences; fetch ID from
server
                    try {
                        UserId = NetworkUtil.getCookie(GET_ID_URI, ""
, "id");
                    } catch (IOException e) {
                        // Catch exceptions such as certification er
rors
                        extMessage = e.toString();
                    }
                    // Store the fetched ID in SharedPreferences
                    sp.edit().putString(ID_KEY, UserId).commit();
                }
                return UserId;
            }

            @Override
            protected void onPostExecute(final String data) {
                String status = (data != null) ? "success" : "error"
;
                Toast.makeText(MainActivity.this, this.getClass().ge
tSimpleName() + " - " + status + " : " +
                    extMessage, Toast.LENGTH_SHORT).show();
            }
        }

        private class SendDataAsyncTack extends AsyncTask<String, Vo
id, Boolean> {

            private String extMessage = "";

            @Override
```

```
        protected Boolean doInBackground(String... params) {

            String url = params[0];
            String id = params[1];
            String nickname = params.length > 2 ? params[2] : nu
ll;
            String imei = params.length > 3 ? params[3] : null;
            Boolean result = false;
            try {
                JSONObject jsonData = new JSONObject();
                jsonData.put(ID_KEY, id);
                if (nickname != null)
                    jsonData.put(NICK_NAME_KEY, nickname);
                if (imei != null)
                    jsonData.put(IMEI_KEY, imei);
                NetworkUtil.sendJSON(url, "", jsonData.toString(
));
                result = true;
            } catch (IOException e) {
                // Catch exceptions such as certification errors
                extMessage = e.toString();
            } catch (JSONException e) {
                extMessage = e.toString();
            }
            return result;
        }

        @Override
        protected void onPostExecute(Boolean result) {
            String status = result ? "Success" : "Error";
            Toast.makeText(MainActivity.this, this.getClass().ge
tSimpleName() + " - " + status + " : " +
                extMessage, Toast.LENGTH_SHORT).show();
        }
    }
}
```

ConfirmFragment.java

```
package org.jssec.android.privacypolicynopreconfirm;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.DialogFragment;
import android.view.LayoutInflater;
```

```
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.TextView;

public class ConfirmFragment extends DialogFragment {

    private DialogListener mListener = null;

    public static interface DialogListener {
        public void onPositiveButtonClick(int type);
        public void onNegativeButtonClick(int type);
    }

    public static ConfirmFragment newInstance(int title, int sen
tence, int type) {
        ConfirmFragment fragment = new ConfirmFragment();
        Bundle args = new Bundle();
        args.putInt("title", title);
        args.putInt("sentence", sentence);
        args.putInt("type", type);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public Dialog onCreateDialog(Bundle args) {
        // *** POINT 1 *** On first launch (or application updat
e), obtain broad consent to transmit user data that will be hand
led by the application.
        final int title = getArguments().getInt("title");
        final int sentence = getArguments().getInt("sentence");
        final int type = getArguments().getInt("type");
        LayoutInflater inflater = (LayoutInflater) getActivity()
.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View content = inflater.inflate(R.layout.fragment_comfir
m, null);
        TextView linkPP = (TextView) content.findViewById(R.id.t
x_link_pp);
        linkPP.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // *** POINT 3 *** Provide methods by which the
user can review the application privacy policy.
                Intent intent = new Intent();
                intent.setClass(getActivity(), WebViewAssetsActi
vity.class);
                startActivity(intent);
            }
        });
        AlertDialog.Builder builder = new AlertDialog.Builder(ge
tActivity());
        builder.setIcon(R.drawable.ic_launcher);
```

```
        builder.setTitle(title);
        builder.setMessage(sentence);
        builder.setView(content);
        builder.setPositiveButton(R.string.buttonConsent, new Di
alogInterface.OnClickListener() {

            public void onClick(DialogInterface dialog, int whic
hButton) {
                if (mListener != null) {
                    mListener.onPositiveButtonClick(type);
                }
            }
        });
        builder.setNegativeButton(R.string.buttonDonotConsent, n
ew DialogInterface.OnClickListener() {

            public void onClick(DialogInterface dialog, int whic
hButton) {
                if (mListener != null) {
                    mListener.onNegativeButtonClick(type);
                }
            }
        });
        Dialog dialog = builder.create();
        dialog.setCanceledOnTouchOutside(false);
        return dialog;
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        if (!(activity instanceof DialogListener)) {
            throw new ClassCastException(activity.toString() + "
 must implement DialogListener.");
        }
        mListener = (DialogListener) activity;
    }

    public void setDialogListener(DialogListener listener) {
        mListener = listener;
    }
}
```

WebViewAssetsActivity.java

```
package org.jssec.android.privacypolicynopreconfirm;

import org.jssec.android.privacypolicynopreconfirm.R;
import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;

public class WebViewAssetsActivity extends Activity {

    // *** POINT 7 *** Place a summary version of the applicatio
n privacy policy in the assets folder
    private final String ABST_PP_URL = "file:///android_asset/Pr
ivacyPolicy/app-policy-abst-privacypolicy-1.0.html";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_webview);
        WebView webView = (WebView) findViewById(R.id.webView);
        WebSettings webSettings = webView.getSettings();
        webSettings.setAllowFileAccess(false);
        webView.loadUrl(ABST_PP_URL);
    }
}
```

5.5.1.3 不需要广泛同意：包含应用隐私策略的应用

要点：

1. 向用户提供查看应用隐私策略的方法。
2. 提供通过用户操作删除传输的数据的方法。
3. 提供通过用户操作停止数据传输的方法
4. 使用 UUID 或 cookie 来跟踪用户数据。
5. 将应用隐私策略的摘要版本放置在素材文件夹中。

MainActivity.java

```
package org.jssec.android.privacypolicynocomprehensive;

import java.io.IOException;
import org.json.JSONException;
import org.json.JSONObject;
import android.os.AsyncTask;
import android.os.Bundle;
import android.content.Intent;
import android.content.SharedPreferences;
import android.support.v4.app.FragmentActivity;
import android.text.Editable;
import android.text.TextWatcher;
```

```
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends FragmentActivity {

    private static final String BASE_URL = "https://www.example.
com/pp";
    private static final String GET_ID_URI = BASE_URL + "/get_id
.php";
    private static final String SEND_DATA_URI = BASE_URL + "/sen
d_data.php";
    private static final String DEL_ID_URI = BASE_URL + "/del_id
.php";
    private static final String ID_KEY = "id";
    private static final String NICK_NAME_KEY = "nickname";
    private static final String PRIVACY_POLICY_PREF_NAME = "priv
acypolicy_preference";
    private String UserId = "";

    private TextWatcher watchHandler = new TextWatcher() {

        @Override
        public void beforeTextChanged(CharSequence s, int start,
int count, int after) {
        }

        @Override
        public void onTextChanged(CharSequence s, int start, int
 before, int count) {
            boolean buttonEnable = (s.length() > 0);
            MainActivity.this.findViewById(R.id.buttonStart).set
Enabled(buttonEnable);
        }

        @Override
        public void afterTextChanged(Editable s) {
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Fetch user ID from serverFetch user ID from server
        new GetDataAsyncTask().execute();
        findViewById(R.id.buttonStart).setEnabled(false);
        ((TextView) findViewById(R.id.editTextNickname)).addText
ChangedListener(watchHandler);
    }
```

```
    public void onSendToServer(View view) {
        String nickname = ((TextView) findViewById(R.id.editText
Nickname)).getText().toString();
        Toast.makeText(MainActivity.this, this.getClass().getSim
pleName() + "¥n - nickname : " + nickname, Toast.LENGTH_SHORT).s
how();
        new sendDataAsyncTack().execute(SEND_DATA_URI, UserId, n
ickname);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_show_pp:
                // *** POINT 1 *** Provide methods by which the
user can review the application privacy policy.
                Intent intent = new Intent();
                intent.setClass(this, WebViewAssetsActivity.clas
s);
                startActivity(intent);
                return true;
            case R.id.action_del_id:
                // *** POINT 2 *** Provide methods by which tran
smitted data can be deleted by user operations.
                new sendDataAsyncTack().execute(DEL_ID_URI, User
Id);
                return true;
            case R.id.action_donot_send_id:
                // *** POINT 3 *** Provide methods by which tran
smitting data can be stopped by user operations.
                // In this sample application if the user data c
annot be sent by user operations,
                // finish the application because we do nothing.
                String message = getString(R.string.stopSendUser
Data);
                Toast.makeText(MainActivity.this, this.getClass(
).getSimpleName() + " - " + message, Toast.LENGTH_SHORT).show();
                finish();
                return true;
        }
        return false;
    }

    private class GetDataAsyncTask extends AsyncTask<String, Void
, String> {

        private String extMessage = "";
```

```
        @Override
        protected String doInBackground(String... params) {
            // *** POINT 4 *** Use UUIDs or cookies to keep trac
k of user data
            // In this sample we use an ID generated on the serv
er side
            SharedPreferences sp = getSharedPreferences(PRIVACY_
POLICY_PREF_NAME, MODE_PRIVATE);
            UserId = sp.getString(ID_KEY, null);
            if (UserId == null) {
                // No token in SharedPreferences; fetch ID from
server
                try {
                    UserId = NetworkUtil.getCookie(GET_ID_URI, ""
, "id");
                } catch (IOException e) {
                    // Catch exceptions such as certification er
rors
                    extMessage = e.toString();
                }
                // Store the fetched ID in SharedPreferences
                sp.edit().putString(ID_KEY, UserId).commit();
            }
            return UserId;
        }

        @Override
        protected void onPostExecute(final String data) {
            String status = (data != null) ? "success" : "error"
;
            Toast.makeText(MainActivity.this, this.getClass().ge
tSimpleName() + " - " + status + " : " +
                extMessage, Toast.LENGTH_SHORT).show();
        }
    }

    private class sendDataAsyncTack extends AsyncTask<String, Vo
id, Boolean> {

        private String extMessage = "";

        @Override
        protected Boolean doInBackground(String... params) {
            String url = params[0];
            String id = params[1];
            String nickname = params.length > 2 ? params[2] : nu
ll;
            Boolean result = false;
            try {
                JSONObject jsonData = new JSONObject();
                jsonData.put(ID_KEY, id);
                if (nickname != null)
```

```
                    jsonData.put(NICK_NAME_KEY, nickname);
                NetworkUtil.sendJSON(url, "", jsonData.toString(
));
                result = true;
            } catch (IOException e) {
                // Catch exceptions such as certification errors
                extMessage = e.toString();
            } catch (JSONException e) {
                extMessage = e.toString();
            }
            return result;
        }

        @Override
        protected void onPostExecute(Boolean result) {
            String status = result ? "Success" : "Error";
            Toast.makeText(MainActivity.this, this.getClass().ge
tSimpleName() + " - " + status + " : " +
                extMessage, Toast.LENGTH_SHORT).show();
        }
    }
}
```

WebViewAssetsActivity.java

```
package org.jssec.android.privacypolicynocomprehensive;

import org.jssec.android.privacypolicynocomprehensive.R;
import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;

public class WebViewAssetsActivity extends Activity {

    // *** POINT 5 *** Place a summary version of the applicatio
n privacy policy in the assets folder
    private static final String ABST_PP_URL = "file:///android_a
sset/PrivacyPolicy/app-policy-abst-privacypolicy-1.0.html";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_webview);
        WebView webView = (WebView) findViewById(R.id.webView);
        WebSettings webSettings = webView.getSettings();
        webSettings.setAllowFileAccess(false);
        webView.loadUrl(ABST_PP_URL);
    }
}
```

5.5.1.4 不包含应用隐私策略的应用

要点：

1. 如果你的应用只使用它在设备中获取的信息，则不需要显示应用隐私策略。
2. 在市场应用或类似应用的文档中，请注意应用不会将其获取的信息传输到外部。

MainActivity.java

```
package org.jssec.android.privacypolicynoinfosent;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GooglePlayServicesClient;
import com.google.android.gms.location.LocationClient;
import android.location.Location;
import android.net.Uri;
import android.os.Bundle;
import android.content.Intent;
import android.content.IntentSender;
import android.support.v4.app.FragmentActivity;
import android.view.Menu;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends FragmentActivity implements Go
oglePlayServicesClient.ConnectionCallbacks,
    GooglePlayServicesClient.OnConnectionFailedListener {

    private LocationClient mLocationClient = null;
    private final int CONNECTION_FAILURE_RESOLUTION_REQUEST = 257
;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mLocationClient = new LocationClient(this, this, this);
    }

    @Override
    protected void onStart() {
        super.onStart();
        // Used to obtain location data
        if (mLocationClient != null) {
            mLocationClient.connect();
        }
    }

    @Override
    protected void onStop() {
```

```
        if (mLocationClient != null) {
            mLocationClient.disconnect();
        }
        super.onStop();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    public void onStartMap(View view) {
        // *** POINT 1 *** You do not need to display an applica
tion privacy policy if your application w
        ill only use the information it obtains within the devic
e.
        if (mLocationClient != null && mLocationClient.isConnect
ed()) {
            Location currentLocation = mLocationClient.getLastLo
cation();
            if (currentLocation != null) {
                Intent intent = new Intent(Intent.ACTION_VIEW, U
ri.parse("geo:" + currentLocation.getLatitude() + "," + currentL
ocation.getLongitude()));
                startActivity(intent);
            }
        }
    }
    @Override
    public void onConnected(Bundle connectionHint) {
        if (mLocationClient != null && mLocationClient.isConnect
ed()) {
            Location currentLocation = mLocationClient.getLastLo
cation();
            if (currentLocation != null) {
                String locationData = "Latitude ¥t: " + currentL
ocation.getLatitude() + "¥n¥tLongitude ¥t: " + currentLocation.g
etLongitude();
                String text = "¥n" + getString(R.string.your_loc
ation_title) + "¥n¥t" + locationData;
                Toast.makeText(MainActivity.this, this.getClass(
).getSimpleName() + text, Toast.LENGTH_SHORT).show();
                TextView appText = (TextView) findViewById(R.id.
appText);
                appText.setText(text);
            }
        }
    }

    @Override
    public void onConnectionFailed(ConnectionResult result) {
        if (result.hasResolution()) {
```

```
            try {
                result.startResolutionForResult(this, CONNECTION
_FAILURE_RESOLUTION_REQUEST);
            } catch (IntentSender.SendIntentException e) {
                e.printStackTrace();
            }
        }
    }

    @Override
    public void onDisconnected() {
        mLocationClient = null;
        Toast.makeText(this, "Disconnected. Please re-connect.",
 Toast.LENGTH_SHORT).show();
    }
}
```

市场上的示例如下。

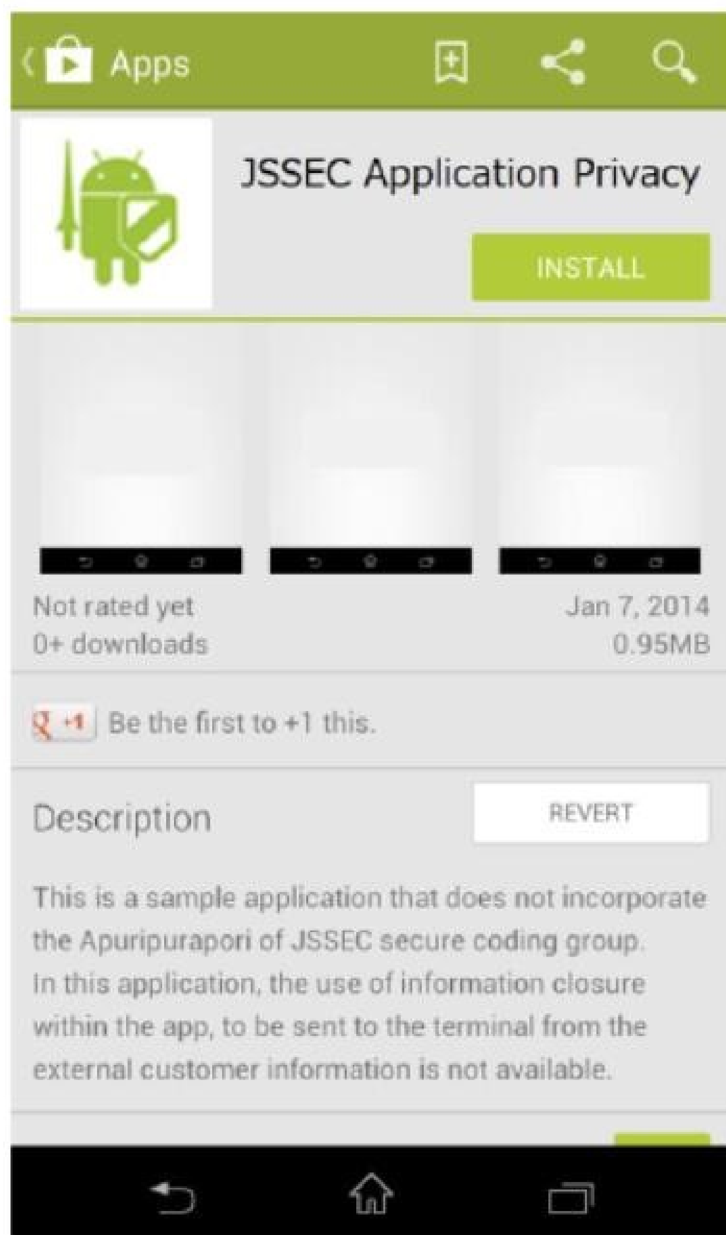


Figure 5.5-3 Description on the marketplace

5.5.2 规则书

处理隐私策略时，遵循以下规则：

5.5.2.1 将用户数据的传输限制为最低需求（必需）

将使用数据传输到外部服务器或其他目标时，将传输限制在提供服务的最低需求。特别是，你应该设计为，应用只能访问这些用户数据，用户可以根据应用描述来想象它们的使用目的。

例如，用户可以想象，它是个警报应用，但不能访问位置数据。另一方面，如果警报应用可以根据用户的位置发出警报，并将其功能写入应用的描述中，则应用可以访问位置数据。

在只需要在应用中访问信息的情况下，避免将信息传输到外部，并采取其他措施来减少无意中泄漏用户数据的可能性。

5.5.2.2 在首次加载（或应用更新）时，获得广泛同意来传输需要特别细致处理或用户可能难以更改的用户数据（必需）

如果应用向外部服务器，传输用户可能难以更改的任何用户数据，或需要特别细致处理的任何用户数据，则应用必须在用户开始使用之前，获得用户的预先同意（选择性加入）- 通知用户哪些类型的信息将被发送到服务器，以及是否会涉及任何第三方厂商。 更具体地说，首次启动时，应用应显示其应用隐私政策并确认该用户已阅读并同意。 此外，无论何时应用更新，通过将新类型的用户数据传输到外部服务器，它都必须再次确认用户已经阅读并同意这些更改。 如果用户不同意，应用应该终止或以其他方式采取措施，来确保所有需要传输数据的功能都被禁用。

这些步骤可以确保，用户了解他们在使用应用时如何处理数据，为用户提供安全感并增强他们对应用的信任。

MainActivity.java

```
protected void onStart() {
    super.onStart();

    // (some portions omitted)

    if (privacyPolicyAgreed <= VERSION_TO_SHOW_COMPREHENSIVE_AGR
EEMENT_ANEW) {
        // *** POINT *** On first launch (or application update)
, obtain broad consent to transmit user data that will be handle
d by the application.
        // When the application is updated, it is only necessary
 to renew the user’s grant of broad consent if the updated appli
cation will handle new types of user data.
        ConfirmFragment dialog = ConfirmFragment.newInstance(
            R.string.privacyPolicy, R.string.agreePrivacyPolicy,
            DIALOG_TYPE_COMPREHENSIVE_AGREEMENT);
        dialog.setDialogListener(this);
        FragmentManager fragmentManager = getSupportFragmentMana
ger();
        dialog.show(fragmentManager, "dialog");
    }
```

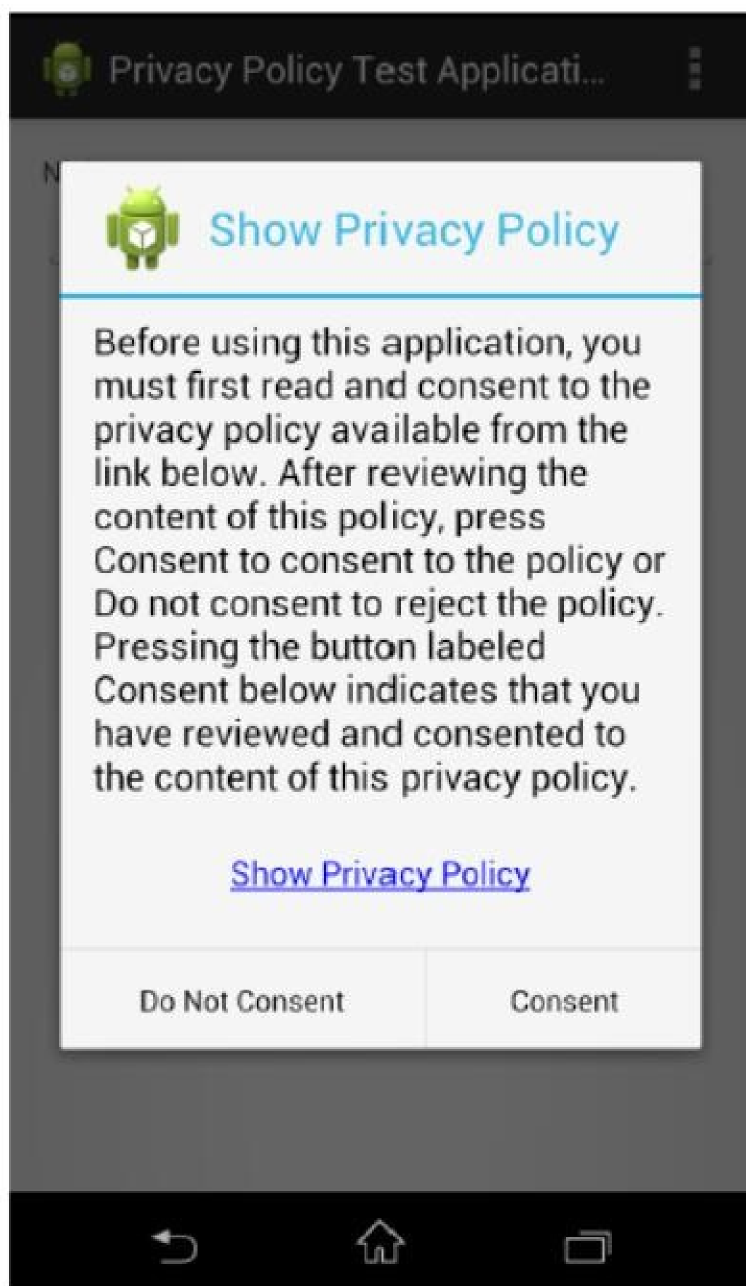


Figure 5.5-4 Example of broad consent

5.5.2.3 在传输需要特殊处理的用户数据之前获得特定的同意（必需）

向外部服务器传输任何需要特别细致处理的用户数据时，除了需要获得一般同意之外，应用必须获得用户对每种这类用户数据（或涉及传输用户数据的每个功能）的预先同意（选择性加入）。 如果用户不同意，则应用不得将相应的数据发送到外部服务器。 这确保用户可以更全面地了解应用的功能（及其提供的服务）和用户对其授予一般同意的，用户数据的传输之间的关系；同时，应用提厂商可以基于更精确的决策，预计获得用户的同意。

MainActivity.java

```
public void onSendToServer(View view) {
    // *** POINT *** Obtain specific consent before transmitting
 user data that requires particularly delicate handling.
    ConfirmFragment dialog = ConfirmFragment.newInstance(R.strin
g.sendLocation, R.string.cofi
    rmSendLocation, DIALOG_TYPE_PRE_CONFIRMATION);
    dialog.setDialogListener(this);
    FragmentManager fragmentManager = getSupportFragmentManager(
);
    dialog.show(fragmentManager, "dialog");
}
```

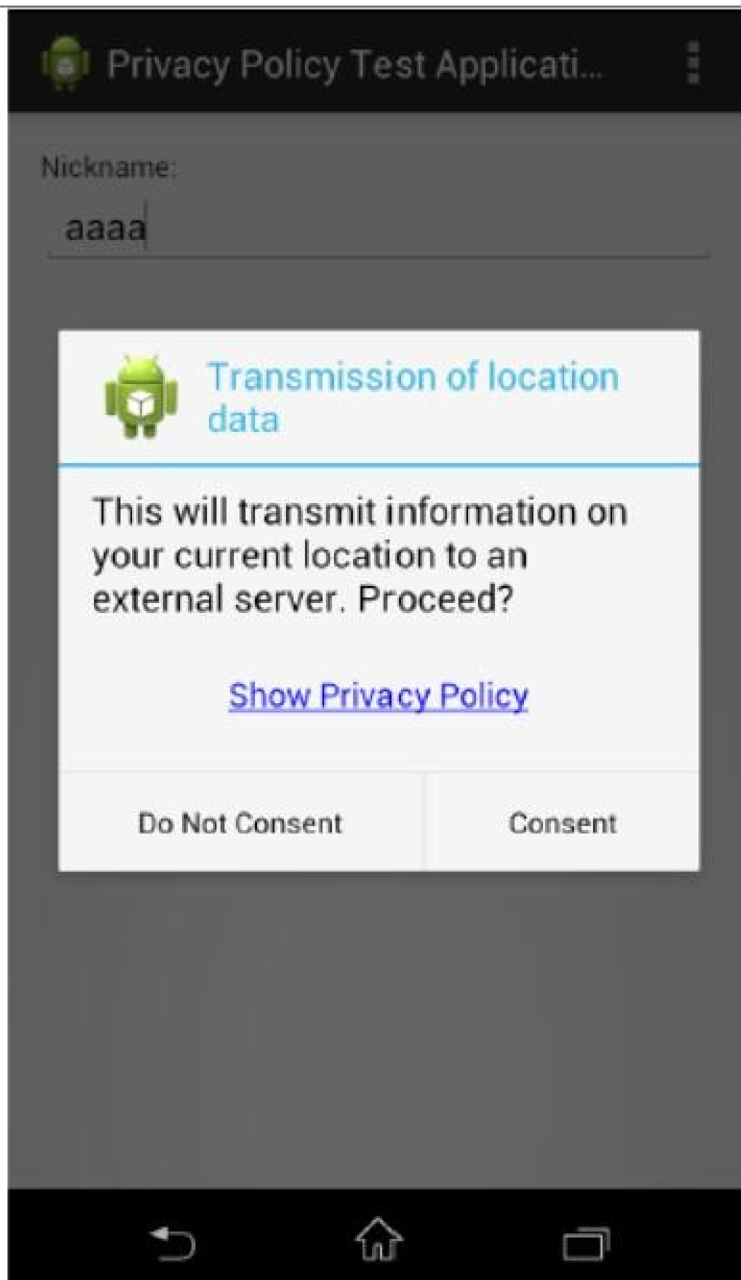


Figure 5.5-5 Example of specific consent

5.5.2.4 向用户提供查看应用隐私策略的方法（必需）

一般来说，Android 应用市场将提供应用隐私策略的链接，供用户在选择安装相应的应用之前进行复查。除了支持此功能之外，应用还需要提供一些方法，用户在设备上安装应用后，可以查看应用隐私策略。特别重要的是提供一些方法，用户可以轻易复查应用隐私政策。在同意的情况下，将用户数据传输到外部服务器来协助用户作出适当决定。

MainActivity.java

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_show_pp:
            // *** POINT *** Provide methods by which the user c
an review the application privacy policy.
            Intent intent = new Intent();
            intent.setClass(this, WebViewAssetsActivity.class);
            startActivity(intent);
            return true;
```

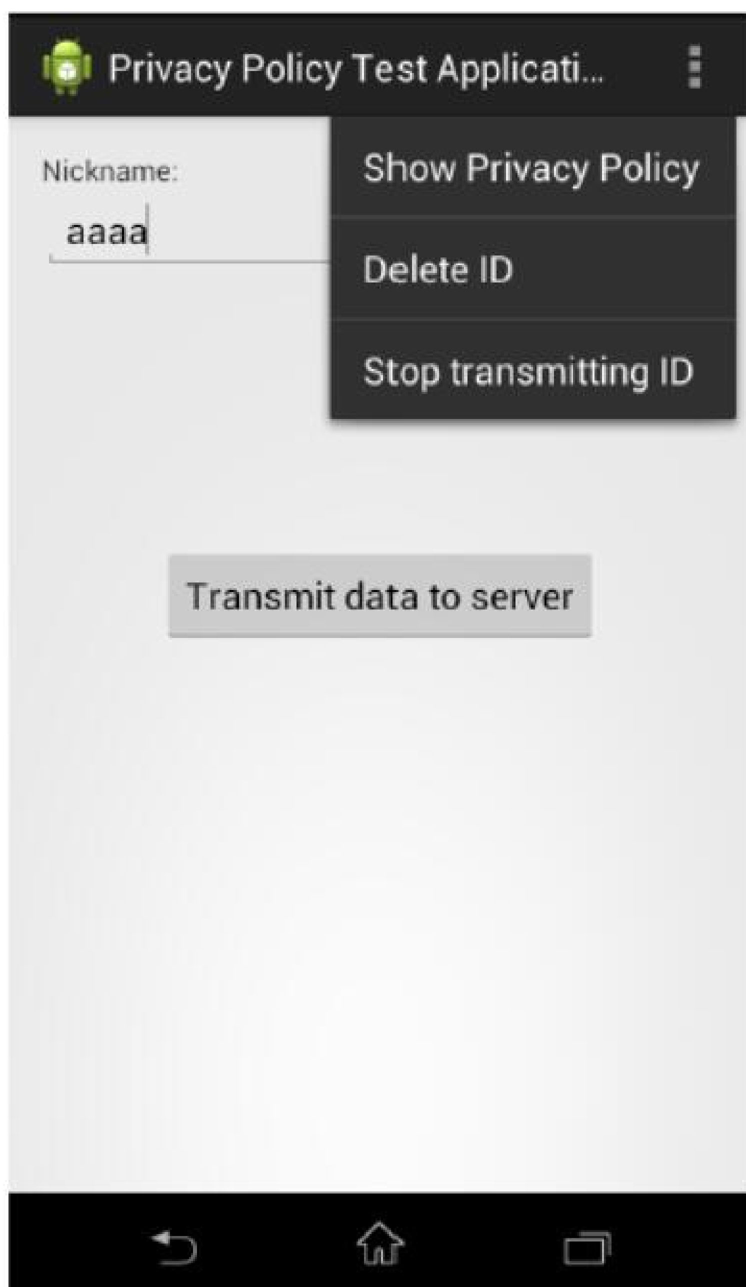


Figure 5.5-6 Context menu to show privacy policy

5.5.2.5 在素材文件夹中放置应用隐私策略的摘要版本（推荐）

将应用隐私策略的摘要版本放在素材文件夹中，来确保用户可以按需对其进行复查，这是一个不错的主意。 确保素材文件夹中存在应用隐私策略，不仅可以让用户随时轻松访问它，还可以避免用户看到由恶意第三方准备的应用隐私策略的伪造或损坏版本的风险。

5.5.2.6 提供可以删除传输的数据的方法，以及可以通过用户操作停止数据传输的方法（推荐）

提供根据用户需要，删除传输到外部服务器的用户数据的方法，是一个好主意。与之相似，在应用本身已经在设备内存储用户数据（或其副本）的情况下，向用户提供用于删除该数据的方法是一个好主意。而且，提供可以根据用户要求停止用户数据发送的方法，是一个好主意。

这一规则（建议）由欧盟推行的“被遗忘权”编纂而成；更普遍的是，在未来，各种提案将要求进一步加强用户保护其数据的权利，这看起来很明显。为此在这些指导方针中，我们建议提供删除用户数据的方法，除非有一些具体原因不能这样做。并且，停止数据传输，主要由浏览器的对应观点“不追踪（否定追踪）”定义。

MainActivity.java

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        (some portions omitted)

        case R.id.action_del_id:
            // *** POINT *** Provide methods by which transmitte
d data can be deleted by user
            operations.
            new SendDataAsyncTack().execute(DEL_ID_URI, UserId);
            return true;
    }
```

5.5.2.7 从 UUID 和 Cookie 中分离设备特定的 ID（推荐）

不应通过与用户数据绑定的方式传输 IMEI 和其他设备特定 ID。 事实上，如果一个设备特定的 ID 和一段用户数据被捆绑在一起，并发布或泄露给公众 - 即使只有一次 - 随后也不可能改变该设备特定的 ID，因此对于把 ID 和用户数据绑定的服务器来说，这是不可能的（或至少 很难）。 在这种情况下，最好使用 UUID 或 cookie（即每次基于随机数重新生成的变量 ID），与用户数据一起传输时代替设备特定的 ID。 这允许实现上面讨论的“被遗忘的权利”的概念。

MainActivity.java

```
@Override
protected String doInBackground(String... params) {
    // *** POINT *** Use UUIDs or cookies to keep track of user
data
    // In this sample we use an ID generated on the server side
    SharedPreferences sp = getSharedPreferences(PRIVACY_POLICY_P
REF_NAME, MODE_PRIVATE);
    UserId = sp.getString(ID_KEY, null);
    if (UserId == null) {
        // No token in SharedPreferences; fetch ID from server
        try {
            UserId = NetworkUtil.getCookie(GET_ID_URI, "", "id")
;
        } catch (IOException e) {
            // Catch exceptions such as certification errors
            extMessage = e.toString();
        }
        // Store the fetched ID in SharedPreferences
        sp.edit().putString(ID_KEY, UserId).commit();
    }
    return UserId;
}
```

5.5.2.8 如果你只在设备内使用用户数据，请通知用户，数据不会传输到外部（推荐）

即使在用户数据只在用户设备中临时访问的情况下，向用户传达这一事实也是一个好主意，来确保用户充分和透明地理解了应用行为。 更具体来说，应该告知用户，应用访问的用户数据只在设备内用于特定的目的，不会被存储或发送。 将此内容传达给用户的可能方法，包括在应用市场上的应用描述中指定它。 仅在设备中临时使用的信息，不需要在应用隐私策略中讨论。

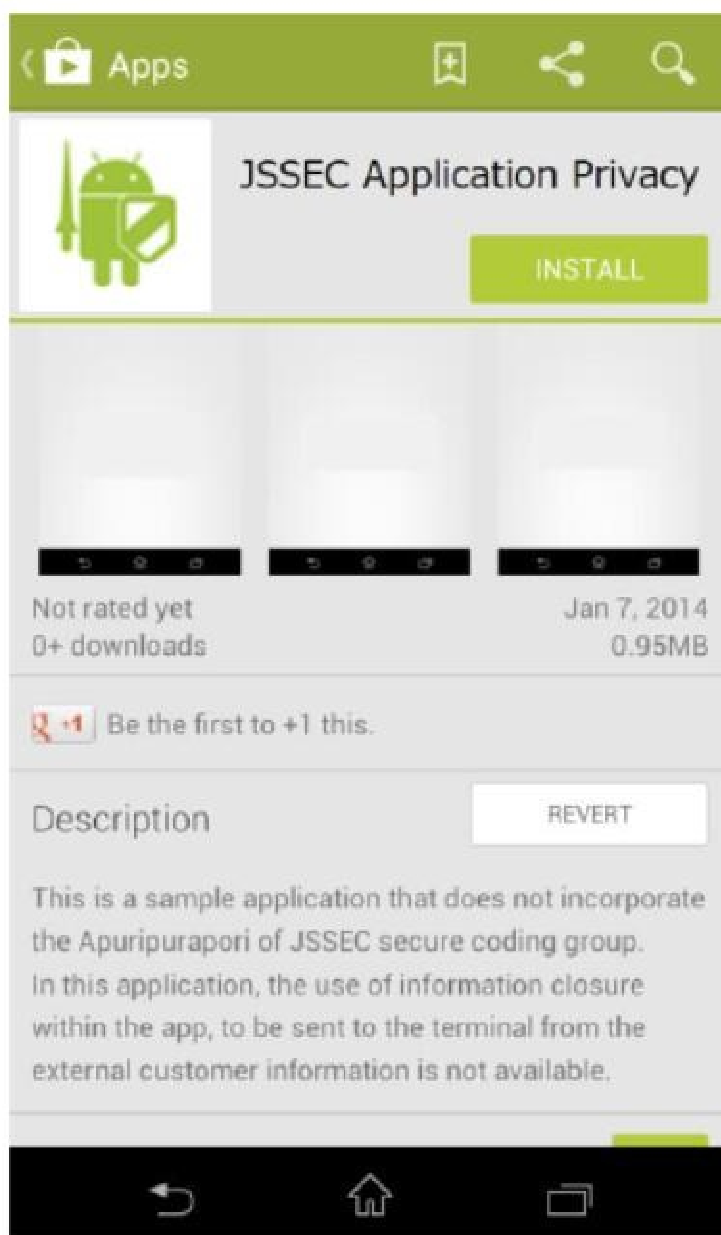


Figure 5.5-7 Description on the marketplace

5.5.3 高级话题

5.3.3.1 隐私政策的背景和上下文

对于智能手机应用获取用户数据，并向外传输该数据的情况，需要准备并显示应用隐私策略，来通知用户一些详细信息，例如收集的数据类型，以及数据被处理的方式。 应包含在应用隐私政策中的内容，在 JMIC SPI 所倡导的 Smartphone Privacy Initiative 中详细说明。 应用隐私策略的主要目标应该是，清楚地声明应用将访问的用户数据的所有类型，数据将用于何种用途，数据将存储在何处以及数据将发送到哪里。

除了应用隐私策略之外，另一个文档是企业隐私策略，它详细说明了公司从各种应用收集的所有用户数据将如何存储，管理和处置。 企业隐私政策对应隐私政策，传统上用于遵循日本个人信息保护法。

准备和展示隐私政策的适当方法的详细说明，以及各种不同类型的隐私政策所起的作用的讨论，可参见文件“对 JSSEC 智能手机创建和展示应用隐私政策的讨论”，可从以下 URL 获得：http://www.jssec.org/event/20140206/03-1_app_policy.pdf（只有日语）。

5.5.3.2 术语表

在下表中，我们定义了这些准则中使用的许多术语；这些定义摘自文件“对 JSSEC 智能手机创建和展示应用隐私政策的讨论”（http://www.jssec.org/event/20140206/03-1_app_policy.pdf）（只有日语）。

术语	描述
企业隐私政策	为保护个人数据而定义的企业政策。 根据日本的个人信息保护法创建。
应用隐私政策	特定于应用的隐私策略。 根据日本内务和通信部（MIC）的智能手机隐私计划（SPI）的指导原则创建。 最好提供摘要，和包含容易理解的解释的详细版本。
应用隐私政策的摘要版本	一份简要文件，简要概述了应用将使用哪些用户信息，用于何种目的，以及这些信息是否会提供给第三方。
应用隐私政策的详细版本	这是一份详细的文件，符合智能手机隐私计划（SPI）和日本总务省（MIC）的智能手机隐私计划 II（SPI II）规定的 8 项内容。
用户易于更改的用户数据	Cookie，UUID，以及其他。
用户难以更改的用户数据	IMEIs, IMSIs, ICCIDs, MAC 地址, OS 生成的 ID, 以及其他。
需要特别处理的用户数据	位置信息，地址本，电话号码，邮箱地址，以及其他。

5.6 密码学

在安全领域，术语“机密性”，“完整性”和“可用性”用于分析对威胁的响应。这三个术语分别指，防止第三方查看私人数据的措施，确保用户引用的数据未被修改的保护措施（或用于检测何时被伪造的技术），以及用户访问服务和数据的能力。在设计安全保护时，所有这三个要素都很重要。特别是，加密技术经常用于确保机密性和完整性，并且 Android 配备了各种加密功能，来允许应用实现机密性和完整性。在本节中，我们将使用示例代码来说明，Android 应用可以安全地执行加密和解密（来确保机密性）和消息认证代码（MAC）或数字签名（来确保完整性）的方法。

5.6.1 示例代码

针对特定用途和条件开发了各种加密方法，包括加密和解密数据（来确保机密性）和检测数据伪造（来确保完整性）等用例。 以下是示例代码，根据每种技术的目的分为三大类加密技术。 在每种情况下，应该能够根据密码技术的特点，选择适当的加密方法和密钥类型。 对于需要更详细考虑的情况，请参见章节“5.6.3.1 选择加密方法”。

在使用加密技术设计实现之前，请务必阅读“5.6.3.3 防止随机数字生成器中的漏洞的措施”。

保护数据免受第三方窃听

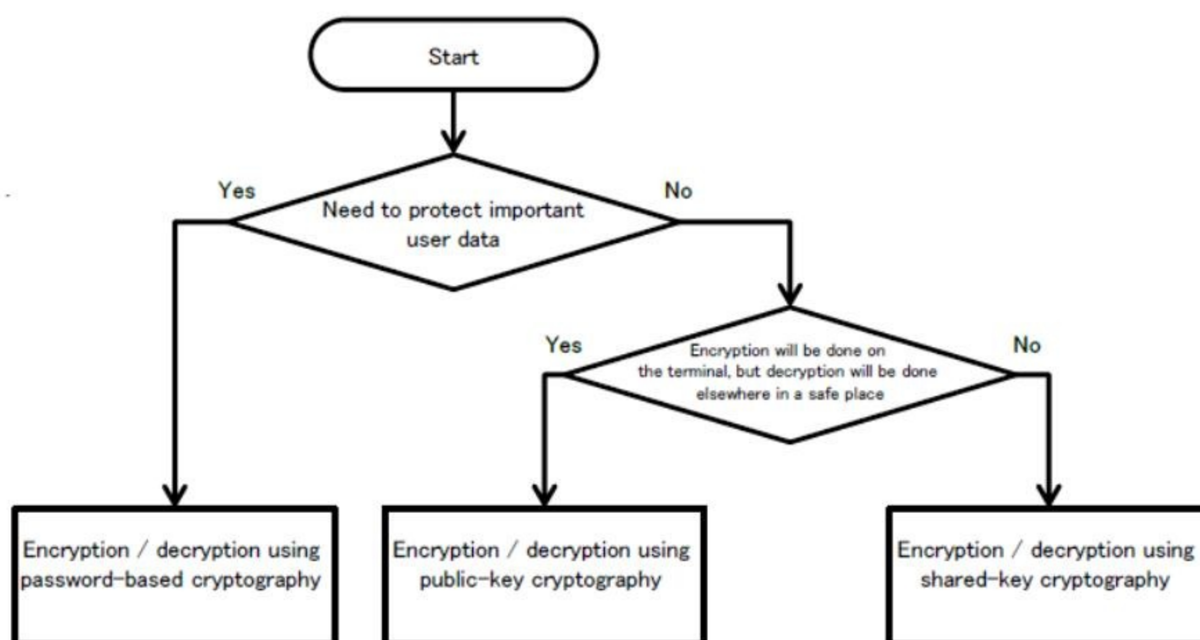


Figure 5.6-1

检测第三方所做的数据伪造

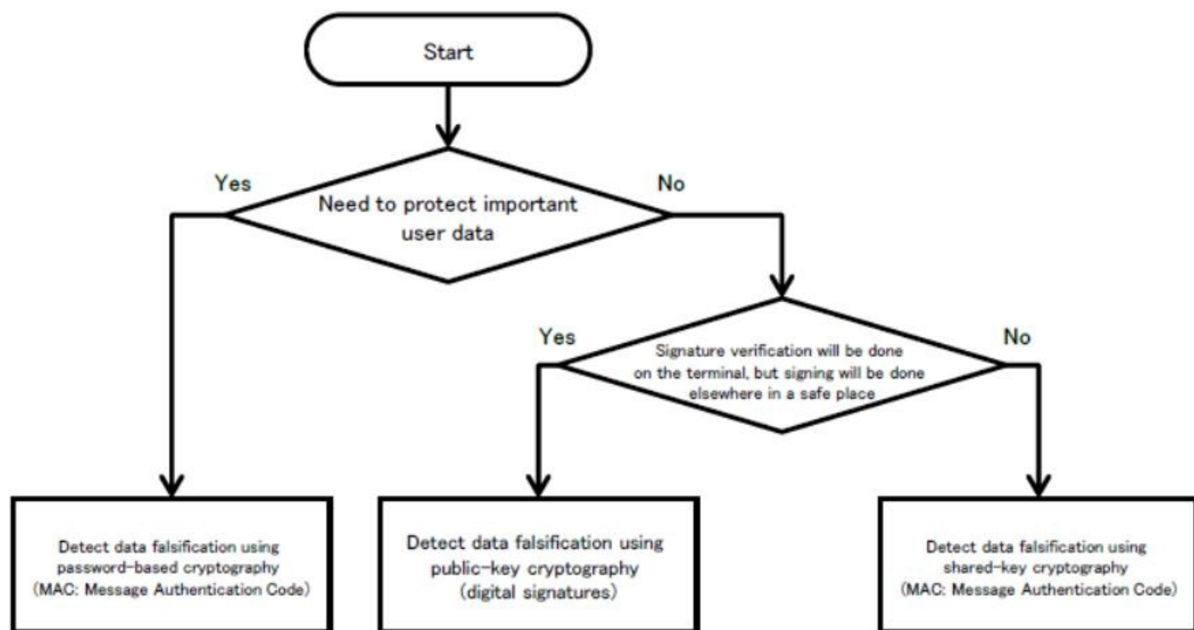


Figure 5.6-2

5.6.1.1 使用基于密码的密钥的加密和解密

你可以使用基于密码的密钥加密，来保护用户的机密数据资产。

要点：

1. 显式指定加密模式和填充。
2. 使用强加密技术（特别是符合相关标准的技术），包括算法，分组加密模式和填充模式。
3. 从密码生成密钥时，使用盐。
4. 从密码生成密钥时，指定适当的哈希迭代计数。
5. 使用足以保证加密强度的密钥长度。

AesCryptoPBEKey.java

```
package org.jssec.android.cryptsymmetricpasswordbasedkey;

import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.spec.InvalidKeySpecException;
import java.util.Arrays;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
```

```
public final class AesCryptoPBEKey {

    // *** POINT 1 *** Explicitly specify the encryption mode an
d the padding.
    // *** POINT 2 *** Use strong encryption technologies (speci
fically, technologies that meet the relevant criteria), includin
g algorithms, block cipher modes, and padding modes.
    // Parameters passed to the getInstance method of the Cipher
 class: Encryption algorithm, block encryption mode, padding rule

    // In this sample, we choose the following parameter values:
 encryption algorithm=AES, block encryption mode=CBC, padding ru
le=PKCS7Padding
    private static final String TRANSFORMATION = "AES/CBC/PKCS7P
adding";
    // A string used to fetch an instance of the class that gene
rates the key
    private static final String KEY_GENERATOR_MODE = "PBEWITHSHA
256AND128BITAES-CBC-BC";
    // *** POINT 3 *** When generating a key from a password, us
e Salt.
    // Salt length in bytes
    public static final int SALT_LENGTH_BYTES = 20;
    // *** POINT 4 *** When generating a key from a password, sp
ecify an appropriate hash iteration count.
    // Set the number of mixing repetitions used when generating
 keys via PBE
    private static final int KEY_GEN_ITERATION_COUNT = 1024;
    // *** POINT 5 *** Use a key of length sufficient to guarant
ee the strength of encryption.
    // Key length in bits
    private static final int KEY_LENGTH_BITS = 128;
    private byte[] mIV = null;
    private byte[] mSalt = null;

    public byte[] getIV() {
        return mIV;
    }

    public byte[] getSalt() {
        return mSalt;
    }

    AesCryptoPBEKey(final byte[] iv, final byte[] salt) {
        mIV = iv;
        mSalt = salt;
    }

    AesCryptoPBEKey() {
        mIV = null;
        initSalt();
    }
```

```
    private void initSalt() {
        mSalt = new byte[SALT_LENGTH_BYTES];
        SecureRandom sr = new SecureRandom();
        sr.nextBytes(mSalt);
    }

    public final byte[] encrypt(final byte[] plain, final char[]
 password) {
        byte[] encrypted = null;
        try {
            // *** POINT 1 *** Explicitly specify the encryption
 mode and the padding.
            // *** POINT 2 *** Use strong encryption technologie
s (specifically, technologies that meet the relevant criteria),
including algorithms, modes, and padding.
            Cipher cipher = Cipher.getInstance(TRANSFORMATION);
            // *** POINT 3 *** When generating keys from passwor
ds, use Salt.
            SecretKey secretKey = generateKey(password, mSalt);
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);
            mIV = cipher.getIV();
            encrypted = cipher.doFinal(plain);
        } catch (NoSuchAlgorithmException e) {
        } catch (NoSuchPaddingException e) {
        } catch (InvalidKeyException e) {
        } catch (IllegalBlockSizeException e) {
        } catch (BadPaddingException e) {
        } finally {
        }
        return encrypted;
    }

    public final byte[] decrypt(final byte[] encrypted, final ch
ar[] password) {
        byte[] plain = null;
        try {
            // *** POINT 1 *** Explicitly specify the encryption
 mode and the padding.
            // *** POINT 2 *** Use strong encryption technologie
s (specifically, technologies that meet the relevant criteria),
including algorithms, block cipher modes, and padding modes.
            Cipher cipher = Cipher.getInstance(TRANSFORMATION);
            // *** POINT 3 *** When generating a key from a pass
word, use Salt.
            SecretKey secretKey = generateKey(password, mSalt);
            IvParameterSpec ivParameterSpec = new IvParameterSpe
c(mIV);
            cipher.init(Cipher.DECRYPT_MODE, secretKey, ivParame
terSpec);
            plain = cipher.doFinal(encrypted);
        } catch (NoSuchAlgorithmException e) {
        } catch (NoSuchPaddingException e) {
```

```
        } catch (InvalidKeyException e) {
        } catch (InvalidAlgorithmParameterException e) {
        } catch (IllegalBlockSizeException e) {
        } catch (BadPaddingException e) {
        } finally {
        }
        return plain;
    }

    private static final SecretKey generateKey(final char[] pass
word, final byte[] salt) {
        SecretKey secretKey = null;
        PBEKeySpec keySpec = null;
        try {
            // *** POINT 2 *** Use strong encryption technologie
s (specifically, technologies that meet the relevant criteria),
including algorithms, block cipher modes, and padding modes.
            // Fetch an instance of the class that generates the
 key
            // In this example, we use a KeyFactory that uses SH
A256 to generate AES-CBC 128-bit keys.
            SecretKeyFactory secretKeyFactory = SecretKeyFactory
.getInstance(KEY_GENERATOR_MODE);
            // *** POINT 3 *** When generating a key from a pass
word, use Salt.
            // *** POINT 4 *** When generating a key from a pass
word, specify an appropriate hash iteration count.
            // *** POINT 5 *** Use a key of length sufficient to
 guarantee the strength of encryption.
            keySpec = new PBEKeySpec(password, salt, KEY_GEN_ITE
RATION_COUNT, KEY_LENGTH_BITS);
            // Clear password
            Arrays.fill(password, '?');
            // Generate the key
            secretKey = secretKeyFactory.generateSecret(keySpec)
;
        } catch (NoSuchAlgorithmException e) {
        } catch (InvalidKeySpecException e) {
        } finally {
        keySpec.clearPassword();
        }
        return secretKey;
    }
}
```

5.6.1.2 使用公钥的加密和解密

在某些情况下，数据加密仅在应用端使用存储的公钥来执行，而解密在单独安全位置（如服务器）在私钥下执行。 在这种情况下，可以使用公钥（非对称密钥）加密。

要点：

1. 显式指定加密模式和填充
2. 使用强加密方法（特别是符合相关标准的技术），包括算法，分组加密模式和填充模式。
3. 使用足以保证加密强度的密钥长度。

RsaCryptoAsymmetricKey.java

```
package org.jssec.android.cryptasymmetrickey;

import java.security.InvalidKeyException;
import java.security.KeyFactory;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.interfaces.RSAPublicKey;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;

public final class RsaCryptoAsymmetricKey {

    // *** POINT 1 *** Explicitly specify the encryption mode an
d the padding.
    // *** POINT 2 *** Use strong encryption methods (specifical
ly, technologies that meet the relevant criteria), including alg
orithms, block cipher modes, and padding modes..
    // Parameters passed to getInstance method of the Cipher cla
ss: Encryption algorithm, block encryption mode, padding rule
    // In this sample, we choose the following parameter values:
 encryption algorithm=RSA, block encryption mode=NONE, padding r
ule=OAEPPADDING.
    private static final String TRANSFORMATION = "RSA/NONE/OAEPP
ADDING";
    // encryption algorithm
    private static final String KEY_ALGORITHM = "RSA";
    // *** POINT 3 *** Use a key of length sufficient to guarant
ee the strength of encryption.
    // Check the length of the key
    private static final int MIN_KEY_LENGTH = 2000;

    RsaCryptoAsymmetricKey() {
    }

    public final byte[] encrypt(final byte[] plain, final byte[]
 keyData) {
        byte[] encrypted = null;
```

```
        try {
            // *** POINT 1 *** Explicitly specify the encryption
 mode and the padding.
            // *** POINT 2 *** Use strong encryption methods (sp
ecifically, technologies that meet the relevant criteria), inclu
ding algorithms, block cipher modes, and padding modes..
            Cipher cipher = Cipher.getInstance(TRANSFORMATION);
            PublicKey publicKey = generatePubKey(keyData);
            if (publicKey != null) {
                cipher.init(Cipher.ENCRYPT_MODE, publicKey);
                encrypted = cipher.doFinal(plain);
            }
        } catch (NoSuchAlgorithmException e) {
        } catch (NoSuchPaddingException e) {
        } catch (InvalidKeyException e) {
        } catch (IllegalBlockSizeException e) {
        } catch (BadPaddingException e) {
        } finally {
        }
        return encrypted;
    }

    public final byte[] decrypt(final byte[] encrypted, final by
te[] keyData) {
        // In general, decryption procedures should be implement
ed on the server side;
        // however, in this sample code we have implemented decr
yption processing within the application to ensure confirmation
of proper execution.
        // When using this sample code in real-world application
s, be careful not to retain any private keys within the applicat
ion.
        byte[] plain = null;
        try {
            // *** POINT 1 *** Explicitly specify the encryption
 mode and the padding.
            // *** POINT 2 *** Use strong encryption methods (sp
ecifically, technologies that meet the relevant criteria), inclu
ding algorithms, block cipher modes, and padding modes..
            Cipher cipher = Cipher.getInstance(TRANSFORMATION);
            PrivateKey privateKey = generatePriKey(keyData);
            cipher.init(Cipher.DECRYPT_MODE, privateKey);
            plain = cipher.doFinal(encrypted);
        } catch (NoSuchAlgorithmException e) {
        } catch (NoSuchPaddingException e) {
        } catch (InvalidKeyException e) {
        } catch (IllegalBlockSizeException e) {
        } catch (BadPaddingException e) {
        } finally {
        }
        return plain;
    }
```

```
    private static final PublicKey generatePubKey(final byte[] k
eyData) {
        PublicKey publicKey = null;
        KeyFactory keyFactory = null;
        try {
            keyFactory = KeyFactory.getInstance(KEY_ALGORITHM);
            publicKey = keyFactory.generatePublic(new X509Encode
dKeySpec(keyData));
        } catch (IllegalArgumentException e) {
        } catch (NoSuchAlgorithmException e) {
        } catch (InvalidKeySpecException e) {
        } finally {
        }
        // *** POINT 3 *** Use a key of length sufficient to gua
rantee the strength of encryption.
        // Check the length of the key
        if (publicKey instanceof RSAPublicKey) {
            int len = ((RSAPublicKey) publicKey).getModulus().bi
tLength();
            if (len < MIN_KEY_LENGTH) {
                publicKey = null;
            }
        }
        return publicKey;
    }

    private static final PrivateKey generatePriKey(final byte[]
keyData) {
        PrivateKey privateKey = null;
        KeyFactory keyFactory = null;
        try {
            keyFactory = KeyFactory.getInstance(KEY_ALGORITHM);
            privateKey = keyFactory.generatePrivate(new PKCS8Enc
odedKeySpec(keyData));
        } catch (IllegalArgumentException e) {
        } catch (NoSuchAlgorithmException e) {
        } catch (InvalidKeySpecException e) {
        } finally {
        }
        return privateKey;
    }
}
```

5.6.1.3 使用预共享密钥的加密和解密

预共享密钥可用于处理大型数据集，或保护应用或用户资产的机密性。

要点：

1. 显式指定加密模式和填充
2. 使用强加密方法（特别是符合相关标准的技术），包括算法，分组加密模式和

填充模式。
3. 使用足以保证加密强度的密钥长度。

AesCryptoPreSharedKey.java

```
package org.jssec.android.cryptsymmetricpresharedkey;

import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public final class AesCryptoPreSharedKey {

    // *** POINT 1 *** Explicitly specify the encryption mode an
d the padding.
    // *** POINT 2 *** Use strong encryption methods (specifical
ly, technologies that meet the relevant cr
    iteria), including algorithms, block cipher modes, and paddi
ng modes.
    // Parameters passed to getInstance method of the Cipher cla
ss: Encryption algorithm, block encryption mode, padding rule
    // In this sample, we choose the following parameter values:
 encryption algorithm=AES, block encryption mode=CBC, padding ru
le=PKCS7Padding
    private static final String TRANSFORMATION = "AES/CBC/PKCS7P
adding";
    // Encryption algorithm
    private static final String KEY_ALGORITHM = "AES";
    // Length of IV in bytes
    public static final int IV_LENGTH_BYTES = 16;
    // *** POINT 3 *** Use a key of length sufficient to guarant
ee the strength of encryption
    // Check the length of the key
    private static final int MIN_KEY_LENGTH_BYTES = 16;
    private byte[] mIV = null;

    public byte[] getIV() {
        return mIV;
    }

    AesCryptoPreSharedKey(final byte[] iv) {
        mIV = iv;
    }

    AesCryptoPreSharedKey() {
```

```
    }

    public final byte[] encrypt(final byte[] keyData, final byte
[] plain) {
        byte[] encrypted = null;
        try {
            // *** POINT 1 *** Explicitly specify the encryption
 mode and the padding.
            // *** POINT 2 *** Use strong encryption methods (sp
ecifically, technologies that meet the relevant criteria), inclu
ding algorithms, block cipher modes, and padding modes.
            Cipher cipher = Cipher.getInstance(TRANSFORMATION);
            SecretKey secretKey = generateKey(keyData);
            if (secretKey != null) {
                cipher.init(Cipher.ENCRYPT_MODE, secretKey);
                mIV = cipher.getIV();
                encrypted = cipher.doFinal(plain);
            }
        } catch (NoSuchAlgorithmException e) {
        } catch (NoSuchPaddingException e) {
        } catch (InvalidKeyException e) {
        } catch (IllegalBlockSizeException e) {
        } catch (BadPaddingException e) {
        } finally {
        }
        return encrypted;
    }

    public final byte[] decrypt(final byte[] keyData, final byte
[] encrypted) {
        byte[] plain = null;
        try {
            // *** POINT 1 *** Explicitly specify the encryption
 mode and the padding.
            // *** POINT 2 *** Use strong encryption methods (sp
ecifically, technologies that meet the relevant criteria), inclu
ding algorithms, block cipher modes, and padding modes.
            Cipher cipher = Cipher.getInstance(TRANSFORMATION);
            SecretKey secretKey = generateKey(keyData);
            if (secretKey != null) {
                IvParameterSpec ivParameterSpec = new IvParamete
rSpec(mIV);
                cipher.init(Cipher.DECRYPT_MODE, secretKey, ivPa
rameterSpec);
                plain = cipher.doFinal(encrypted);
            }
        } catch (NoSuchAlgorithmException e) {
        } catch (NoSuchPaddingException e) {
        } catch (InvalidKeyException e) {
        } catch (InvalidAlgorithmParameterException e) {
        } catch (IllegalBlockSizeException e) {
        } catch (BadPaddingException e) {
        } finally {
```

```
        }
        return plain;
    }

    private static final SecretKey generateKey(final byte[] keyD
ata) {
        SecretKey secretKey = null;
        try {
            // *** POINT 3 *** Use a key of length sufficient to
 guarantee the strength of encryption
            if (keyData.length >= MIN_KEY_LENGTH_BYTES) {
                // *** POINT 2 *** Use strong encryption methods
 (specifically, technologies that meet the relevant criteria), i
ncluding algorithms, block cipher modes, and padding modes.
                secretKey = new SecretKeySpec(keyData, KEY_ALGOR
ITHM);
            }
        } catch (IllegalArgumentException e) {
        } finally {
        }
        return secretKey;
    }
}
```

5.6.1.4 使用基于密码的密钥来检测数据伪造

你可以使用基于密码的（共享密钥）加密来验证用户数据的完整性。

要点：

1. 显式指定加密模式和填充。
2. 使用强加密方法（特别是符合相关标准的技术），包括算法，分组加密模式和填充模式。
3. 从密码生成密钥时，使用盐。
4. 从密码生成密钥时，指定适当的哈希迭代计数。
5. 使用足以保证 MAC 强度的密钥长度。

HmacPBEKey.java

```
package org.jssec.android.signsymmetricpasswordbasedkey;

import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.spec.InvalidKeySpecException;
import java.util.Arrays;
import javax.crypto.Mac;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
```

```
public final class HmacPBEKey {

    // *** POINT 1 *** Explicitly specify the encryption mode an
d the padding.
    // *** POINT 2 *** Use strong encryption methods (specifical
ly, technologies that meet the relevant criteria), including alg
orithms, block cipher modes, and padding modes.
    // Parameters passed to the getInstance method of the Mac cl
ass: Authentication mode
    private static final String TRANSFORMATION = "PBEWITHHMACSHA
1";
    // A string used to fetch an instance of the class that gene
rates the key
    private static final String KEY_GENERATOR_MODE = "PBEWITHHMA
CSHA1";
    // *** POINT 3 *** When generating a key from a password, us
e Salt.
    // Salt length in bytes
    public static final int SALT_LENGTH_BYTES = 20;
    // *** POINT 4 *** When generating a key from a password, sp
ecify an appropriate hash iteration count.
    // Set the number of mixing repetitions used when generating
 keys via PBE
    private static final int KEY_GEN_ITERATION_COUNT = 1024;
    // *** POINT 5 *** Use a key of length sufficient to guarant
ee the MAC strength.
    // Key length in bits
    private static final int KEY_LENGTH_BITS = 160;
    private byte[] mSalt = null;

    public byte[] getSalt() {
        return mSalt;
    }

    HmacPBEKey() {
        initSalt();
    }

    HmacPBEKey(final byte[] salt) {
        mSalt = salt;
    }

    private void initSalt() {
        mSalt = new byte[SALT_LENGTH_BYTES];
        SecureRandom sr = new SecureRandom();
        sr.nextBytes(mSalt);
    }

    public final byte[] sign(final byte[] plain, final char[] pa
ssword) {
        return calculate(plain, password);
    }
```

```
    private final byte[] calculate(final byte[] plain, final char
[] password) {
        byte[] hmac = null;
        try {
            // *** POINT 1 *** Explicitly specify the encryption
 mode and the padding.
            // *** POINT 2 *** Use strong encryption methods (sp
ecifically, technologies that meet the relevant criteria), inclu
ding algorithms, block cipher modes, and padding modes.
            Mac mac = Mac.getInstance(TRANSFORMATION);
            // *** POINT 3 *** When generating a key from a pass
word, use Salt.
            SecretKey secretKey = generateKey(password, mSalt);
            mac.init(secretKey);
            hmac = mac.doFinal(plain);
        } catch (NoSuchAlgorithmException e) {
        } catch (InvalidKeyException e) {
        } finally {
        }
        return hmac;
    }

    public final boolean verify(final byte[] hmac, final byte[]
plain, final char[] password) {
        byte[] hmacForPlain = calculate(plain, password);
        if (Arrays.equals(hmac, hmacForPlain)) {
            return true;
        }
        return false;
    }

    private static final SecretKey generateKey(final char[] pass
word, final byte[] salt) {
        SecretKey secretKey = null;
        PBEKeySpec keySpec = null;
        try {
            // *** POINT 2 *** Use strong encryption methods (sp
ecifically, technologies that meet the relevant criteria), inclu
ding algorithms, block cipher modes, and padding modes.
            // Fetch an instance of the class that generates the
 key
            // In this example, we use a KeyFactory that uses SH
A1 to generate AES-CBC 128-bit keys.
            SecretKeyFactory secretKeyFactory = SecretKeyFactory
.getInstance(KEY_GENERATOR_MODE);
            // *** POINT 3 *** When generating a key from a pass
word, use Salt.
            // *** POINT 4 *** When generating a key from a pass
word, specify an appropriate hash iteration count.
            // *** POINT 5 *** Use a key of length sufficient to
 guarantee the MAC strength.
            keySpec = new PBEKeySpec(password, salt, KEY_GEN_ITE
RATION_COUNT, KEY_LENGTH_BITS);
```

```
            // Clear password
            Arrays.fill(password, '?');
            // Generate the key
            secretKey = secretKeyFactory.generateSecret(keySpec)
;
        } catch (NoSuchAlgorithmException e) {
        } catch (InvalidKeySpecException e) {
        } finally {
            keySpec.clearPassword();
        }
        return secretKey;
    }
}
```

5.6.1.5 使用公钥来检测数据伪造

所处理的数据的签名，由存储在不同的安全位置（如服务器）中的私钥确定时，你可以使用公钥（不对称密钥）加密来处理涉及应用端公钥存储的应用，出于验证数据签名的目的。

要点：

1. 显式指定加密模式和填充。
2. 使用强加密方法（特别是符合相关标准的技术），包括算法，分组加密模式和填充模式。
3. 使用足以保证签名强度的密钥长度。

RsaSignAsymmetricKey.java

```
package org.jssec.android.signasymmetrickey;

import java.security.InvalidKeyException;
import java.security.KeyFactory;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.Signature;
import java.security.SignatureException;
import java.security.interfaces.RSAPublicKey;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;

public final class RsaSignAsymmetricKey {

    // *** POINT 1 *** Explicitly specify the encryption mode an
d the padding.
    // *** POINT 2 *** Use strong encryption methods (specifical
ly, technologies that meet the relevant criteria), including alg
orithms, block cipher modes, and padding modes.
```

```
    // Parameters passed to the getInstance method of the Cipher
 class: Encryption algorithm, block encryption mode, padding rule

    // In this sample, we choose the following parameter values:
 encryption algorithm=RSA, block encryption mode=NONE, padding r
ule=OAEPPADDING.
    private static final String TRANSFORMATION = "SHA256withRSA"
;
    // encryption algorithm
    private static final String KEY_ALGORITHM = "RSA";
    // *** POINT 3 *** Use a key of length sufficient to guarant
ee the signature strength.
    // Check the length of the key
    private static final int MIN_KEY_LENGTH = 2000;

    RsaSignAsymmetricKey() {
    }

    public final byte[] sign(final byte[] plain, final byte[] ke
yData) {
        // In general, signature procedures should be implemente
d on the server side;
        // however, in this sample code we have implemented sign
ature processing within the application to ensure confirmation o
f proper execution.
        // When using this sample code in real-world application
s, be careful not to retain any private keys within the applicat
ion.
        byte[] sign = null;
        try {
            // *** POINT 1 *** Explicitly specify the encryption
 mode and the padding.
            // *** POINT 2 *** Use strong encryption methods (sp
ecifically, technologies that meet the relevant criteria), inclu
ding algorithms, block cipher modes, and padding modes.
            Signature signature = Signature.getInstance(TRANSFOR
MATION);
            PrivateKey privateKey = generatePriKey(keyData);
            signature.initSign(privateKey);
            signature.update(plain);
            sign = signature.sign();
        } catch (NoSuchAlgorithmException e) {
        } catch (InvalidKeyException e) {
        } catch (SignatureException e) {
        } finally {
        }
        return sign;
    }

    public final boolean verify(final byte[] sign, final byte[]
plain, final byte[] keyData) {
        boolean ret = false;
        try {
```

```
            // *** POINT 1 *** Explicitly specify the encryption
 mode and the padding.
            // *** POINT 2 *** Use strong encryption methods (sp
ecifically, technologies that meet the relevant criteria), inclu
ding algorithms, block cipher modes, and padding modes.
            Signature signature = Signature.getInstance(TRANSFOR
MATION);
            PublicKey publicKey = generatePubKey(keyData);
            signature.initVerify(publicKey);
            signature.update(plain);
            ret = signature.verify(sign);
        } catch (NoSuchAlgorithmException e) {
        } catch (InvalidKeyException e) {
        } catch (SignatureException e) {
        } finally {
        }
        return ret;
    }

    private static final PublicKey generatePubKey(final byte[] k
eyData) {
        PublicKey publicKey = null;
        KeyFactory keyFactory = null;
        try {
            keyFactory = KeyFactory.getInstance(KEY_ALGORITHM);
            publicKey = keyFactory.generatePublic(new X509Encode
dKeySpec(keyData));
        } catch (IllegalArgumentException e) {
        } catch (NoSuchAlgorithmException e) {
        } catch (InvalidKeySpecException e) {
        } finally {
        }
        // *** POINT 3 *** Use a key of length sufficient to gua
rantee the signature strength.
        // Check the length of the key
        if (publicKey instanceof RSAPublicKey) {
            int len = ((RSAPublicKey) publicKey).getModulus().bi
tLength();
            if (len < MIN_KEY_LENGTH) {
                publicKey = null;
            }
        }
        return publicKey;
    }

    private static final PrivateKey generatePriKey(final byte[]
keyData) {
        PrivateKey privateKey = null;
        KeyFactory keyFactory = null;
        try {
            keyFactory = KeyFactory.getInstance(KEY_ALGORITHM);
            privateKey = keyFactory.generatePrivate(new PKCS8Enc
odedKeySpec(keyData));
```

```
        } catch (IllegalArgumentException e) {
        } catch (NoSuchAlgorithmException e) {
        } catch (InvalidKeySpecException e) {
        } finally {
        }
        return privateKey;
    }
}
```

5.6.1.6 使用预共享密钥来检测数据伪造

你可以使用预共享密钥来验证应用资产或用户资产的完整性。

要点：

1. 显式指定加密模式和填充。
2. 使用强加密方法（特别是符合相关标准的技术），包括算法，分组加密模式和填充模式。
3. 使用足以保证 MAC 强度的密钥长度。

HmacPreSharedKey.java

```
package org.jssec.android.signsymmetricpresharedkey;

import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import javax.crypto.Mac;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;

public final class HmacPreSharedKey {

    // *** POINT 1 *** Explicitly specify the encryption mode an
d the padding.
    // *** POINT 2 *** Use strong encryption methods (specifical
ly, technologies that meet the relevant criteria), including alg
orithms, block cipher modes, and padding modes.
    // Parameters passed to the getInstance method of the Mac cl
ass: Authentication mode
    private static final String TRANSFORMATION = "HmacSHA256";
    // Encryption algorithm
    private static final String KEY_ALGORITHM = "HmacSHA256";
    // *** POINT 3 *** Use a key of length sufficient to guarant
ee the MAC strength.
    // Check the length of the key
    private static final int MIN_KEY_LENGTH_BYTES = 16;

    HmacPreSharedKey() {
    }
```

```
    public final byte[] sign(final byte[] plain, final byte[] ke
yData) {
        return calculate(plain, keyData);
    }

    public final byte[] calculate(final byte[] plain, final byte
[] keyData) {
        byte[] hmac = null;
        try {
            // *** POINT 1 *** Explicitly specify the encryption
 mode and the padding.
            // *** POINT 2 *** Use strong encryption methods (sp
ecifically, technologies that meet the relevant criteria), inclu
ding algorithms, block cipher modes, and padding modes.
            Mac mac = Mac.getInstance(TRANSFORMATION);
            SecretKey secretKey = generateKey(keyData);
            if (secretKey != null) {
                mac.init(secretKey);
                hmac = mac.doFinal(plain);
            }
        } catch (NoSuchAlgorithmException e) {
        } catch (InvalidKeyException e) {
        } finally {
        }
        return hmac;
    }

    public final boolean verify(final byte[] hmac, final byte[]
plain, final byte[] keyData) {
    byte[] hmacForPlain = calculate(plain, keyData);
        if (hmacForPlain != null && Arrays.equals(hmac, hmacForP
lain)) {
            return true;
        }
        return false;
    }

    private static final SecretKey generateKey(final byte[] keyD
ata) {
        SecretKey secretKey = null;
        try {
            // *** POINT 3 *** Use a key of length sufficient to
 guarantee the MAC strength.
            if (keyData.length >= MIN_KEY_LENGTH_BYTES) {
            // *** POINT 2 *** Use strong encryption methods (sp
ecifically, technologies that meet the relevant criteria), inclu
ding algorithms, block cipher modes, and padding modes.
            secretKey = new SecretKeySpec(keyData, KEY_ALGORITHM
);
            }
        } catch (IllegalArgumentException e) {
        } finally {
```

```
        }
        return secretKey;
    }
}
```

5.6.2 规则书

使用加密技术时，遵循以下规则：

5.6.2.1 指定加密算法时，请显式指定加密模式和填充（必需）

在使用加密技术和数据验证等密码学技术时，加密模式和填充必须显式指定。 在 Android 应用开发中使用加密时，你将主要使用`java.crypto`中的`Cipher`类。 为了使用`Cipher`类，你将首先通过指定要使用的加密类型，来创建`Cipher`类对象的实例。 这个指定被称为转换，并且有两种格式可以指定转换：

- 算法/模式/填充
- 算法

在后一种情况下，加密模式和填充将隐式设置为 Android 可以访问的加密服务供应器的适当默认值。 这些默认值优先考虑便利性和兼容性而选择，并且在某些情况下可能不是特别安全的选择。 为此，为了确保正确的安全保护，必须使用两种格式中的前者，其中显式指定了加密模式和填充。

5.6.2.2 使用强算法（特别是符合相关标准的算法）（必需）

使用加密技术时，选择符合特定标准的强算法很重要。 此外，在算法允许多个密钥长度的情况下，重要的是要考虑应用的整个产品生命周期，并选择足以确保安全性的密钥长度。 此外，对于一些加密模式和填充模式，存在已知的攻击策略；对这些威胁做出有力的选择是非常重要的。

确实，选择弱加密方法会造成灾难性后果。 例如，被加密来防止第三方窃听的文件，实际上可能仅受到无效保护，并且可能允许第三方窃听。 由于 IT 的不断进步导致加密分析技术的持续改进，因此至关重要的是，考虑并选择一个算法，它能够在运行的整个期间，保证安全性。在此时间，你希望应用保持运行。

实际加密技术的标准因国家而异，详见下表（单位：位）。

表 5.6-1 NIST(USA) NIST SP800-57

算法生命周期	对称密钥加密	非对称密钥加密	椭圆曲线加密	**HASH**（数字签名）	**HASH**（随机数生成）
~2010	80	1024	160	160	160
~2030	112	2048	224	224	160
2030~	128	3072	256	256	160

表 5.6-2 ECRYPT II (EU)

算法生命周期	对称密钥加密	非对称密钥加密	椭圆曲线加密	HASH
2009~2012	80	1248	160	160
2009~2020	96	1776	192	192
2009~2030	112	2432	224	224
2009~2040	128	3248	256	256
2009~	256	15424	512	512

表 5.6-3 CRYPTREC(Japan) CRYPTREC 加密算法列表

技术族		名称
公钥加密	签名	DSA,ECDSA,RSA-PSS,RSASSA-PKCS1-V1_5
	机密性	RSA-OAEP
	密钥共享	DH,ECDH
共享密钥加密	64 位块加密	3-key Triple DES
	128 位块加密	AES,Camellia
	流式加密	KCipher-2
哈希函数		SHA-256,SHA-384,SHA-512
加密使用模式	密文模式	CBC,CFB,CTR,OFB
	认证密文模式	CCM,GCM
消息认证代码		CMAC,HMAC
实体认证		ISO/IEC 9798-2,ISO/IEC 9798-3

5.6.2.3 使用基于密码的加密时，不要在设备上存储密码（必需）

在基于密码的加密中，当根据用户输入的密码生成加密密钥时，请勿将密码存储在设备中。 基于密码的加密的优点是无需管理加密密钥；将密码存储在设备上消除了这一优势。 无需多说，在设备上存储密码会产生其他应用窃听的风险，因此出于安全原因，在设备上存储密码也是不可接受的。

5.6.2.4 从密码生成密钥时，使用盐（必需）

在基于密码的加密中，当根据用户输入的密码生成加密密钥时，请始终使用盐。 另外，如果你要在同一设备中为不同用户提供功能，请为每个用户使用不同的盐。 原因是，如果你仅使用简单的哈希函数生成加密密钥而不使用盐，则可以使用称为“彩虹表”的技术轻松恢复密码。使用了盐时，会使用相同的密码生成的密钥 将是不同的（不同的哈希值），防止使用彩虹表来搜索密钥。

示例：

```
public final byte[] encrypt(final byte[] plain, final char[] pas
sword) {
    byte[] encrypted = null;
    try {
        // *** POINT *** Explicitly specify the encryption mode
and the padding.
        // *** POINT *** Use strong encryption methods (specific
ally, technologies that meet the relevant criteria), including a
lgorithms, block cipher modes, and padding modes.
        Cipher cipher = Cipher.getInstance(TRANSFORMATION);
        // *** POINT *** When generating keys from passwords, us
e Salt.
        SecretKey secretKey = generateKey(password, mSalt);
```

5.6.2.5 从密码生成密钥时，指定适当的哈希迭代计数（必需）

在基于密码的加密中，当根据用户输入的密码生成加密密钥时，你需要选择在密钥生成过程（“拉伸”）中，散列过程的重复次数；指定足够大的数字来确保安全性非常重要。一般来说，1,000 或更大的迭代次数是足够的。如果你使用密钥来保护更有价值的资产，请指定 1,000,000 或更高的计数。由于散列函数的单个计算所需的处理时间很少，因此攻击者可能很容易进行爆破攻击。因此，通过使用拉伸方法（其中散列处理重复多次），我们可以有意确保该过程消耗大量时间，因此爆破攻击的成本更高。请注意，拉伸重复次数也会影响应用的处理速度，因此请谨慎选择合适的值。

示例：

```
private static final SecretKey generateKey(final char[] password
, final byte[] salt) {
    SecretKey secretKey = null;
    PBEKeySpec keySpec = null;

    (Omit)

    // *** POINT *** When generating a key from password, use Sa
lt.
    // *** POINT *** When generating a key from password, specif
y an appropriate hash iteration count.
    // *** POINT *** Use a key of length sufficient to guarantee
 the strength of encryption.
    keySpec = new PBEKeySpec(password, salt, KEY_GEN_ITERATION_C
OUNT, KEY_LENGTH_BITS);
```

5.6.2.6 采取措施来增加密码强度（推荐）

在基于密码的加密中，当基于用户输入的密码生成加密密钥时，生成的密钥的强度受用户密码强度的强烈影响，因此值得采取措施来加强从用户那里收到的密码。 例如，你可以要求密码长度至少为 8 个字符，并且包含多种类型的字符 - 可能至少包含一个字母，一个数字和一个符号。

5.6.3 高级话题

5.6.3.1 选择加密方法

在上面的示例代码中，我们展示了三种加密方法的实现示例，每种加密方法用于加密解密以及数据伪造的检测。 你可以使用“图 5.6-1”，“图 5.6-2”，根据你的应用粗略选择使用哪种加密方法。 另一方面，加密方法的更加精细的选择，需要更详细地比较各种方法的特征。 在下面我们考虑一些这样的比较。

用于加密和解密的密码学方法的比较

公钥密码术具有很高的处理成本，因此不适合大规模数据处理。但是，因为用于加密和解密的密钥不同，所以仅仅在应用侧处理公钥（即，只执行加密），并且在不同（安全）位置执行解密的情况下，管理密钥相对容易。共享密钥加密是一种通用的加密方案，但限制很少，但在这种情况下，相同的密钥用于加密和解密，因此有必要将密钥安全地存储在应用中，从而使密钥管理变得困难。基于密码的密钥系统（基于密码的共享密钥系统）通过用户指定的密码生成密钥，避免了在设备中存储密钥相关的密码的需求。此方法用于仅仅保护用户资产，但不保护应用资产的应用。由于加密强度取决于密码强度，因此有必要选择密码，其复杂度与要保护的资产价值成比例增长。请参阅“5.6.2.6 采取措施来增加密码强度（推荐）”。

表 5.6-4 用于加密和解密的密码学方法的比较

条目/加密方法	公钥	共享密钥	基于密码
处理大规模数据	否（开销太大）	OK	OK
保护应用（或服务）资产	OK	OK	否（允许用户窃取）
保护用户资产	OK	OK	OK
加密强度	取决于密钥长度	取决于密钥长度	取决于密码强度，盐和哈希重复次数
密钥存储	简单（仅公钥）	困难	简单
由应用执行的过程	加密（解密在服务器或其它地方完成）	加密和解密	加密和解密

用于检测数据伪造的密码学方法的比较

这里的比较与上面讨论的加密和解密类似，除了与数据大小对应的条目不再相关。

表 5.6-5 用于检测数据伪造的密码学方法的比较

条目/加密方法	公钥	共享密钥	基于密码
保护应用（或服务）资产	OK	OK	否（允许用户伪造）
保护用户资产	OK	OK	OK
加密强度	取决于密钥长度	取决于密钥长度	取决于密码强度，盐和哈希重复次数
密钥存储	简单（仅公钥）	困难，请参考“5.6.3.4 保护密钥”	简单
由应用执行的过程	签名验证（签名在服务器或其它地方完成）	MAC 计算和验证	MAC 计算和验证

MAC：消息认证代码

请注意，这些准则主要关注被视为低级或中级资产的资产保护，根据“3.1.3 资产分类和保护对策”一节中讨论的分类。 由于使用加密涉及的问题，比其他预防性措施（如访问控制）更多，如密钥存储问题，因此只有资产不能在 Android 操作系统安全模式下有效保护时，才应该考虑加密。

5.6.3.2 随机数的生成

使用加密技术时，选择强加密算法和加密模式，以及足够长的密钥，来确保应用和服务处理的数据的安全性，这非常重要。 然而，即使所有这些选择都做得适当，当形成安全协议关键的密钥被泄漏或猜测时，所使用的算法所保证的安全强度立即下降为零。

即使对于在AES和类似协议下，用于共享密钥加密的初始向量（IV），或者用于基于密码的加密的盐，较大偏差也可以使第三方轻松发起攻击，从而增加数据泄漏或污染的风险 。 为了防止这种情况，有必要以第三方难以猜测它们的值的方式，产生密钥和 IV，而随机数在确保这一必要实现的方面，起着非常重要的作用。 产生随机数的设备称为随机数生成器。 尽管硬件随机数生成器（RNG）可能使用传感器或其他设备，通过测量无法预测或再现的自然现象来产生随机数，但更常见的是用软件实现的随机数生成器，称为伪随机数生成器（PRNG）。

在Android应用中，可以通过 `SecureRandom` 类生成用于加密的足够安全的随机数。 `SecureRandom` 类的功能由一个称为 `Provider` 的实现提供。 多个供应器（实现）可以在内部存在，并且如果没有明确指定供应器，则会选择默认供应器。 出于这个原因，也可以在不知道供应器存在的情况下，使用 `SecureRandom` 来实现。 在下面,我们提供的例子演示了如何使用 `SecureRandom` 。

请注意，根据 Android 版本的不同， `SecureRandom` 可能存在一些缺陷，需要在实施中采取预防措施。 请参阅“5.6.3.3 防止随机数生成器中的漏洞的措施”。

使用 `SecureRandom` （默认实现）

```
import java.security.SecureRandom;

[...]

SecureRandom random = new SecureRandom();
byte[] randomBuf = new byte [128];
random.nextBytes(randomBuf);

[...]
```

使用 `SecureRandom` （明确的特定算法）

```
import java.security.SecureRandom;

[...]

SecureRandom random = SecureRandom.getInstance("SHA1PRNG");
byte[] randomBuf = new byte [128];
random.nextBytes(randomBuf);

[...]
```

使用 `SecureRandom` （明确的特定实现（供应器））

```
import java.security.SecureRandom;

[...]

SecureRandom random = SecureRandom.getInstance("SHA1PRNG", “Cryp
to”);
byte[] randomBuf = new byte [128];
random.nextBytes(randomBuf);

[...]
```

程序中发现的伪随机数发生器，例如 `SecureRandom` ，通常基于一些基本过程来操作，如“图 5.6-3 伪随机数发生器的内部过程”中所述。 输入一个随机数种子来初始化内部状态；此后，每次生成随机数时更新内部状态，从而允许生成随机数序列。

随机数种子

种子在伪随机数发生器（PRNG）中起着非常重要的作用。 如上所述，PRNG 必须通过指定种子来初始化。 此后，用于生成随机数的过程是确定性算法，因此如果指定相同的种子，则会得到相同的随机数序列。 这意味着如果第三方获得（即窃听）或猜测 PRNG 的种子，他可以产生相同的随机数序列，从而破坏随机数提供的机密性和完整性属性。

出于这个原因，随机数生成器的种子本身就是一个高度机密的信息 - 而且必须以无法预测或猜测的方式来选择。 例如，不应使用时间信息或设备特定数据（例如 MAC 地址，IMEI 或 Android ID）来构建 RNG 种子。 在许多 Android 设备上， `/dev/urandom` 或 `/dev/random` 可用，Android 提供的 `SecureRandom` 默认实现使用这些设备文件，来确定随机数生成器的种子。 就机密性而言，只要 RNG 种子仅存在于内存中，除获得 root 权限的恶意软件工具外，几乎没有由第三方发现的风险。 如果你需要实现，即使在已 root 的设备上仍然有效的安全措施，请咨询安全设计和实现方面的专家。

伪随机数生成器的内部状态

伪随机数发生器的内部状态由种子初始化，然后在每次生成随机数时更新。 就像由相同种子初始化的 PRNG 一样，具有相同内部状态的两个 PRNG 随后将产生完全相同的随机数序列。 因此，保护内部状态免受第三方窃听也很重要。 但是，由于内部状态存在于内存中，除了拥有 root 访问权的恶意软件工具外，几乎没有发现任何第三方的风险。 如果你需要实现，即使在已 root 的设备上仍然有效的安全措施，请咨询安全设计和实现方面的专家。

5.6.3.3 防范随机数生成器中的漏洞的措施

在 Android 4.3.x 及更早版本中发现， `SecureRandom` 的 `Crypto` 供应器实现拥有内部状态熵（随机性）不足的缺陷。 特别是在 Android 4.1.x 及更早版本中， `Crypto` 供应器是 `SecureRandom` 的唯一可用实现，因此大多数直接或间接使用 `SecureRandom` 的应用都受此漏洞影响。 同样，Android 4.2 和更高版本中，作为 `SecureRandom` 的默认实现而提供的 `AndroidOpenSSL` 供应器拥有这个缺陷，由 `OpenSSL` 使用的作为随机数种子的大部分数据在应用之间共享（Android 4.2.x-4.3 .x），产生了一个漏洞，任何应用都可以轻松预测其他应用生成的随机数。 下表详细说明了各种 Android OS 版本中存在的漏洞的影响。

表 5.6-6 Android操作系统版本和受到每个漏洞的影响的功能

Android OS/漏洞	`SecureRandom` 的 `Crypto` 供应器实现的内部状态熵不足	可以猜测其他程序中 `OpenSSL` 所使用的随机数
4.1.x 及之前	`SecureRandom` 的默认实现，`Crypto` 供应器的显式使用，由 `Cipher` 类提供的加密功能，HTTPS 通信功能等	无影响
4.2 - 4.3.x	使用明确标识的 `Crypto` 供应器	`SecureRandom` 的默认实现，Android OpenSSL 供应器的显式使用， `OpenSSL` 提供的随机数生成功能的直接使用，由 `Cipher` 类提供的加密功能，HTTPS 通信功能等
4.4 及之后	无影响	无影响

自 2013 年 8 月以来，Google 已经向其合作伙伴（设备制造商等），分发了用于消除这些 Android 操作系统漏洞的补丁。但是，与 `SecureRandom` 相关的这些漏洞影响了广泛的应用，包括加密功能和 HTTPS 通信功能，并且据推测许多设备仍未修补。 因此，在设计针对 Android 4.3.x 和更早版本的应用时，我们建议你采纳以下站点中讨论的对策（实现）。

http://android-developers.blogspot.jp/2013/08/some-securerandom-thoughts.html

5.6.3.4 密钥保护

使用加密技术来确保敏感数据的安全性（机密性和完整性）时，只要密钥本身的数据内容是可用的，即使最健壮的加密算法和密钥长度，也不能保护数据免受第三方攻击。 出于这个原因，正确处理密钥是使用加密时需要考虑的最重要的项目之一。当然，根据你尝试保护的资产的级别，正确处理密钥可能需要非常复杂的设计和实现技术，这些技术超出了本指南的范围。 在这里，我们只能提供一些基本想法，有关安全处理各种应用和密钥的存储位置;我们的讨论没有扩展到特定的实现方法，并且必要时我们建议你咨询安全设计和实现方面的专家。

首先，“图 5.6-4 加密密钥的位置和保护它们的策略”，说明了 Android 智能手机和平板电脑中，用于储存密钥和相关用途的各种位置，并概述了保护它们的策略。

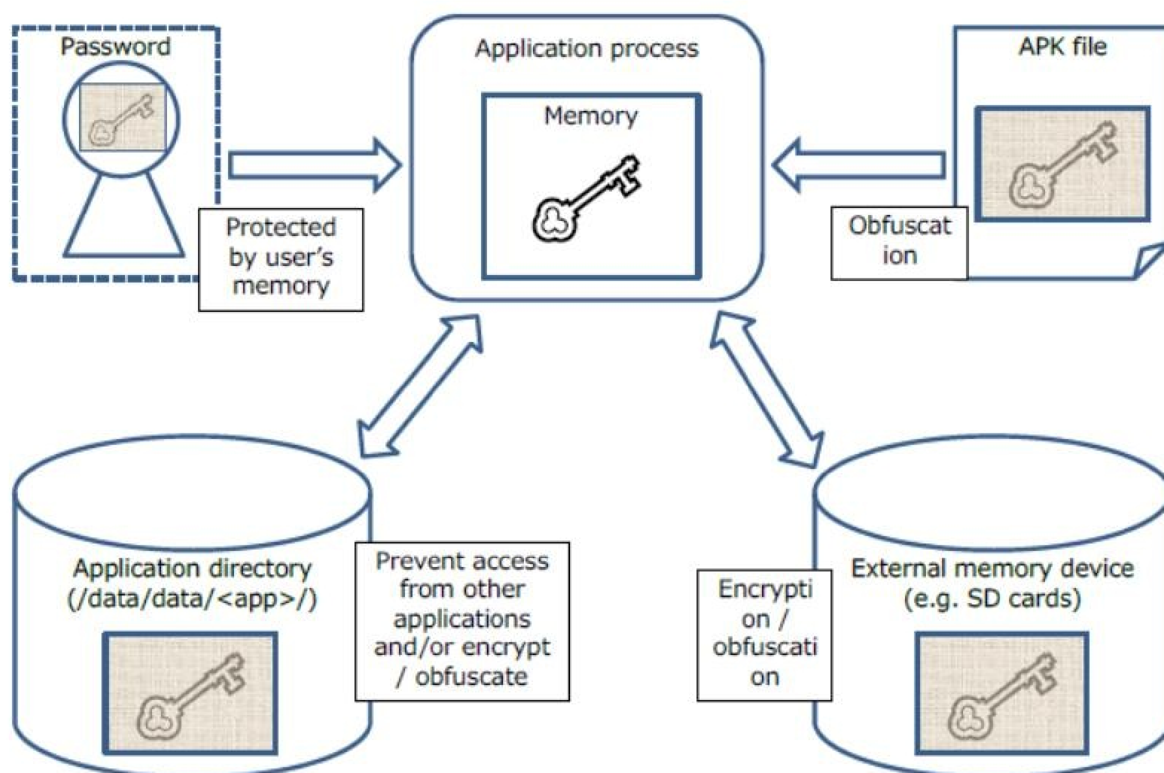


Figure 5.6-4 Places of encrypt keys and strategies for protecting them.

下表总结了受密钥保护的资产的资产类别，以及适用于各种资产所有者的保护策略。 资产类别的更多信息，请参阅“3.1.3 资产分类和保护对策”。

表 5.6-7 资产分类和保护对策

资产所有者	设备用户		应用/服务供应者	
资产级别	高	中低	高	中低
密钥储存位置		保护策略		
用户内存	提高密码强度		不允许使用用户密码	
应用目录（非公共存储）	密钥加密或混淆	禁止来自应用外部的读写操作	密钥加密或混淆	禁止来自应用外部的读写操作
APK 文件		混淆密钥数据。注：要注意大多数 Java 混淆工具，例如 Proguard，不会混淆数据字符串。		
SD 卡或者其它（公共存储）		加密或混淆密钥数据		

在下文中，我们讨论适用于存储密钥的各个地方的保护措施。

储存在用户内存中的密钥

这里我们考虑基于密码的加密。 从密码生成密钥时，密钥存储位置是用户内存，因此不存在由于恶意软件而造成泄漏的危险。 但是，根据密码的强度，可能很容易重现密钥。 出于这个原因，有必要采取步骤来确保密码的强度， 类似于让用户指定服务登录密码时采取的步骤；例如，密码可能受到 UI 的限制，或者可能会使用警告消息。 请参阅“5.6.2.6 采取措施增加密码的强度（推荐）”。 当然，当密码存储在用户用户中时，必须记住密码将被遗忘的可能性。 为确保在忘记密码的情况下可以恢复数据，必须将备份数据存储在设备以外的安全位置（例如服务器上）。

储存在应用目录中的密钥

当密钥以私有模式，存储在应用目录中时，密钥数据不能被其他应用读取。 另外，如果应用禁用备份功能，用户也将无法访问数据。 因此，当存储用于保护应用资产的密钥时，应该禁用备份。

但是，如果你还需要针对使用 root 权限的应用或用户保护密钥，则必须对密钥进行加密或混淆。 对于用于保护用户资产的密钥，你可以使用基于密码的加密。 对于用于加密应用资产的密钥，你希望这些资产对于用户是不可见的，你必须将用于资产加密的密钥存储在 APK 文件中，并且必须对密钥数据进行混淆处理。

储存在 APK 文件中的密钥

由于可以访问APK文件中的数据，因此通常这不适合存储机密数据（如密钥）。 在 APK 文件中存储密钥时，你必须对密钥数据进行混淆处理，并采取措施确保数据无法轻易从 APK 文件中读取。

储存在公共存储位置（例如 SD 卡）的密钥

由于公共存储可以被所有应用访问，因此通常它不适合存储机密数据（如密码）。 将密钥存储在公共位置时，需要对密钥数据进行加密或混淆处理，来确保无法轻易访问数据。 另请参阅上面的“存储在应用目录中的密钥”中提出的保护措施，来了解还必须针对具有 root 权限的应用或用户来保护密钥。

在进程内存中处理密钥

使用 Android 中可用的加密技术时，必须在加密过程之前，在上图中所示的应用进程以外的地方，对加密或混淆的密钥数据进行解密（或者，对于基于密码的密钥，则需要生成密钥）。在这种情况下，密钥数据将以未加密的形式驻留在进程内存中。另一方面，应用的内存通常不会被其他应用读取，因此如果资产类别位于这些准则涵盖的范围内，则没有采取特定步骤来确保安全性的特别需求。在密钥数据以未加密的形式出现（即使它们以这种方式存在于进程内存中）是不可接受的的情况下，由于特定目标或由应用处理的资产级别，可能有必要对密钥数据和加密逻辑，采取混淆处理或其他技术。但是，这些方法在 Java 层面上难以实现；相反，你将在 JNI 层面上使用混淆工具。这些措施不在本准则的范围之内；咨询安全设计和实现方面的专家。

5.6.3.5 通过 Google Play 服务解决安全供应器的漏洞

Google Play 服务（5.0 和更高版本）提供了一个称为供应器安装器的框架，可用于解决安全供应器中的漏洞。

首先，安全提供应器提供了基于 Java 密码体系结构（JCA）的各种加密相关的算法的实现。 这些安全供应器算法可以通过诸如 `Cipher` ， `Signature` 和 `Mac` 等类来使用，来在 Android 应用中使用加密技术。 一般来说，只要在加密技术相关的实现中发现漏洞，就需要快速响应。 事实上，以恶意目的利用这些漏洞可能会导致严重损害。 由于加密技术也与安全供应器相关，所以希望用于解决漏洞的修订越快越好。

执行安全供应器修订的最常见方法是使用设备更新。通过设备更新执行修订的过程，起始于设备制造商准备更新，之后用户将此更新应用于其设备。因此，应用是否可以访问安全供应器的最新版本（包括最新版本），实际上取决于制造商和用户的遵从性。相反，使用来自 Google Play 服务的供应器安装器，可确保应用可以访问自动更新的安全供应器版本。

使用来自 Google Play 服务的供应器安装器，通过从应用调用供应器安装器，可以访问由 Google Play 服务提供的安全供应器。 Google Play 服务会通过 Google Play 商店自动更新，因此供应器安装器所提供的安全供应器，将自动更新到最新版本，而不依赖制造商或用户的遵从性。

调用供应器安装器的示例代码如下所示。

调用供应器安装器

```
import com.google.android.gms.common.GooglePlayServicesUtil;
import com.google.android.gms.security.ProviderInstaller;

public class MainActivity extends Activity
    implements ProviderInstaller.ProviderInstallListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ProviderInstaller.installIfNeededAsync(this, this);
        setContentView(R.layout.activity_main);
    }

    @Override
    public void onProviderInstalled() {
        // Called when Security Provider is the latest version,
or when installation completes
    }

    @Override
    public void onProviderInstallFailed(int errorCode, Intent re
coveryIntent) {
        GoogleApiAvailability.getInstance().showErrorNotificatio
n(this, errorCode);
    }
}
```

5.7 使用指纹认证功能

目前正在研究和开发的各种用于生物认证的方法中，使用面部信息和声音特征的方法尤其突出。在这些方法中，使用指纹认证来识别个体的方法自古以来就有所使用，并且今天被用于签名（通过拇指印）和犯罪调查等目的。指纹识别的应用也在计算机世界的几个领域中得到了发展，并且近年来，这些方法已经开始作为高度便利的技术（提供诸如易于输入的优点）而享有广泛认可，用于一些领域，例如识别智能手机的物主（主要用于解锁屏幕）。

在这些趋势下，Android 6.0（API Level 23）在终端上整合了指纹认证框架，允许应用使用指纹认证功能来识别个人身份。在下面我们将讨论一些使用指纹认证时要记住的安全预防措施。

5.7.1 示例代码

下面我们提供示例代码，来允许应用使用 Android 的指纹认证功能。

要点：

1. 声明使用 `USE_FINGERPRINT` 权限
2. 从 `AndroidKeyStore` 供应器获取实例
3. 通知用户需要指纹注册才能创建密钥
4. 创建（注册）密钥时，请使用没有漏洞的加密算法（符合标准）
5. 创建（注册）密钥时，启用用户（指纹）认证请求（不要指定启用认证的持续时间）
6. 设计你的应用的前提是，指纹注册的状态将在密钥创建和使用密钥期间发生变化
7. 将加密数据限制为，可通过指纹认证以外的方法恢复（替换）的项东西

MainActivity.java

```
package authentication.fingerprint.android.jssec.org.fingerprint
authentication;

import android.app.AlertDialog;
import android.hardware.fingerprint.FingerprintManager;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Base64;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
```

```
public class MainActivity extends AppCompatActivity {

    private FingerprintAuthentication mFingerprintAuthentication
;
    private static final String SENSITIVE_DATA = "sensitive data"
;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mFingerprintAuthentication = new FingerprintAuthenticati
on(this);
        Button button_fingerprint_auth = (Button) findViewById(R
.id.button_fingerprint_auth);
        button_fingerprint_auth.setOnClickListener(new View.OnCl
ickListener() {

            @Override
            public void onClick(View v) {
                if (!mFingerprintAuthentication.isAuthenticating
()) {
                    if (authenticateByFingerprint()) {
                        showEncryptedData(null);
                        setAuthenticationState(true);
                    }
                } else {
                    mFingerprintAuthentication.cancel();
                }
            }
        });
    }

    private boolean authenticateByFingerprint() {
        if (!mFingerprintAuthentication.isFingerprintHardwareDet
ected()) {
            // Terminal is not equipped with a fingerprint sensor

            return false;
        }
        if (!mFingerprintAuthentication.isFingerprintAuthAvailab
le()) {
            // *** POINT 3 *** Notify users that fingerprint reg
istration will be required to create a key
            new AlertDialog.Builder(this)
                .setTitle(R.string.app_name)
                .setMessage("No fingerprint information has been
 registered.¥n" +
                    "Click ¥"Security¥" on the Settings menu to
register fingerprints. ¥n" +
                    "Registering fingerprints allows easy authen
tication.")
                .setPositiveButton("OK", null)
```

```
                .show();
            return false;
        }
        // Callback that receives the results of fingerprint aut
hentication
        FingerprintManager.AuthenticationCallback callback = new
 FingerprintManager.AuthenticationCallback() {

            @Override
            public void onAuthenticationError(int errorCode, Cha
rSequence errString) {
                showMessage(errString, R.color.colorError);
                reset();
            }

            @Override
            public void onAuthenticationHelp(int helpCode, CharS
equence helpString) {
                showMessage(helpString, R.color.colorHelp);
            }

            @Override
            public void onAuthenticationSucceeded(FingerprintMan
ager.AuthenticationResult result) {
                Cipher cipher = result.getCryptoObject().getCiph
er();
                try {
                    // *** POINT 7*** Restrict encrypted data to
 items that can be restored (replaced) by methods other than fin
gerprint authentication
                    byte[] encrypted = cipher.doFinal(SENSITIVE_
DATA.getBytes());
                    showEncryptedData(encrypted);
                } catch (IllegalBlockSizeException | BadPaddingE
xception e) {
                }
                showMessage(getString(R.string.fingerprint_auth_
succeeded), R.color.colorAuthenticated);
                reset();
            }
            @Override
            public void onAuthenticationFailed() {
                showMessage(getString(R.string.fingerprint_auth_
failed), R.color.colorError);
            }
        };
        if (mFingerprintAuthentication.startAuthentication(callb
ack)) {
            showMessage(getString(R.string.fingerprint_processin
g), R.color.colorNormal);
            return true;
        }
        return false;
```

```
    }

    private void setAuthenticationState(boolean authenticating)
{
        Button button = (Button) findViewById(R.id.button_finger
print_auth);
        button.setText(authenticating ? R.string.cancel : R.stri
ng.authenticate);
    }

    private void showEncryptedData(byte[] encrypted) {
        TextView textView = (TextView) findViewById(R.id.encrypt
edData);
        if (encrypted != null) {
            textView.setText(Base64.encodeToString(encrypted, 0)
);
        } else {
            textView.setText("");
        }
    }

    private String getCurrentTimeString() {
        long currentTimeMillis = System.currentTimeMillis();
        Date date = new Date(currentTimeMillis);
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat(
"HH:mm:ss.SSS");
        return simpleDateFormat.format(date);
    }

    private void showMessage(CharSequence msg, int colorId) {
        TextView textView = (TextView) findViewById(R.id.textVie
w);
        textView.setText(getCurrentTimeString() + " :¥n" + msg);
        textView.setTextColor(getResources().getColor(colorId, n
ull));
    }

    private void reset() {
        setAuthenticationState(false);
    }
}
```

FingerprintAuthentication.java

```
package authentication.fingerprint.android.jssec.org.fingerprint
authentication;

import android.app.KeyguardManager;
import android.content.Context;
import android.hardware.fingerprint.FingerprintManager;
```

```
import android.os.CancellationSignal;
import android.security.keystore.KeyGenParameterSpec;
import android.security.keystore.KeyInfo;
import android.security.keystore.KeyPermanentlyInvalidatedExcept
ion;
import android.security.keystore.KeyProperties;
import java.io.IOException;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.KeyStore;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.UnrecoverableKeyException;
import java.security.cert.CertificateException;
import java.security.spec.InvalidKeySpecException;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;

public class FingerprintAuthentication {

    private static final String KEY_NAME = "KeyForFingerprintAut
hentication";
    private static final String PROVIDER_NAME = "AndroidKeyStore"
;
    private KeyguardManager mKeyguardManager;
    private FingerprintManager mFingerprintManager;
    private CancellationSignal mCancellationSignal;
    private KeyStore mKeyStore;
    private KeyGenerator mKeyGenerator;
    private Cipher mCipher;

    public FingerprintAuthentication(Context context) {
        mKeyguardManager = (KeyguardManager) context.getSystemSe
rvice(Context.KEYGUARD_SERVICE);
        mFingerprintManager = (FingerprintManager) context.getSy
stemService(Context.FINGERPRINT_SERVICE
        );
        reset();
    }

    public boolean startAuthentication(final FingerprintManager.
AuthenticationCallback callback) {
        if (!generateAndStoreKey())
            return false;
        if (!initializeCipherObject())
            return false;
        FingerprintManager.CryptoObject cryptoObject = new Finge
rprintManager.CryptoObject(mCipher);
        mCancellationSignal = new CancellationSignal();
```

```
        // Callback to receive the results of fingerprint authen
tication
        FingerprintManager.AuthenticationCallback hook = new Fin
gerprintManager.AuthenticationCallback() {

            @Override
            public void onAuthenticationError(int errorCode, Cha
rSequence errString) {
                if (callback != null)
                    callback.onAuthenticationError(errorCode, er
rString);
                reset();
            }

            @Override
            public void onAuthenticationHelp(int helpCode, CharS
equence helpString) {
                if (callback != null)
                    callback.onAuthenticationHelp(helpCode, help
String);
            }

            @Override
            public void onAuthenticationSucceeded(FingerprintMan
ager.AuthenticationResult result) {
                if (callback != null)
                    callback.onAuthenticationSucceeded(result);
                reset();
            }

            @Override
            public void onAuthenticationFailed() {
                if (callback != null)
                    callback.onAuthenticationFailed();
            }
        };
        // Execute fingerprint authentication
        mFingerprintManager.authenticate(cryptoObject, mCancella
tionSignal, 0, hook, null);
        return true;
    }

    public boolean isAuthenticating() {
        return mCancellationSignal != null && !mCancellationSign
al.isCanceled();
    }

    public void cancel() {
        if (mCancellationSignal != null) {
            if (!mCancellationSignal.isCanceled())

                mCancellationSignal.cancel();
        }
```

```
    }

    private void reset() {
        try {
            // *** POINT 2 *** Obtain an instance from the "Andr
oidKeyStore" Provider
            mKeyStore = KeyStore.getInstance(PROVIDER_NAME);
            mKeyGenerator = KeyGenerator.getInstance(KeyProperti
es.KEY_ALGORITHM_AES, PROVIDER_NAME);
            mCipher = Cipher.getInstance(KeyProperties.KEY_ALGOR
ITHM_AES
                + "/" + KeyProperties.BLOCK_MODE_CBC
                + "/" + KeyProperties.ENCRYPTION_PADDING_PKCS7);
        } catch (KeyStoreException | NoSuchPaddingException
            | NoSuchAlgorithmException | NoSuchProviderException
 e) {
            throw new RuntimeException("failed to get cipher ins
tances", e);
        }
        mCancellationSignal = null;
    }

    public boolean isFingerprintAuthAvailable() {
        return (mKeyguardManager.isKeyguardSecure()
            && mFingerprintManager.hasEnrolledFingerprints()) ?
true : false;
    }

    public boolean isFingerprintHardwareDetected() {
        return mFingerprintManager.isHardwareDetected();
    }

    private boolean generateAndStoreKey() {
        try {
            mKeyStore.load(null);
            if (mKeyStore.containsAlias(KEY_NAME))
                mKeyStore.deleteEntry(KEY_NAME);
            mKeyGenerator.init(
                // *** POINT 4 *** When creating (registering) k
eys, use an encryption algorithm that is not vulnerable (meets s
tandards)
                new KeyGenParameterSpec.Builder(KEY_NAME, KeyPro
perties.PURPOSE_ENCRYPT)
                    .setBlockModes(KeyProperties.BLOCK_MODE_CBC)
                    .setEncryptionPaddings(KeyProperties.ENCRYPT
ION_PADDING_PKCS7)
                // *** POINT 5 *** When creating (registering) k
eys, enable requests for user (fingerprint) authentication (do n
ot specify the duration over which authentication is enabled)
                    .setUserAuthenticationRequired(true)
                    .build());
            // Generate a key and store it in Keystore(AndroidKe
yStore)
```

```
                mKeyGenerator.generateKey();
                return true;
            } catch (IllegalStateException e) {
                return false;
            } catch (NoSuchAlgorithmException | InvalidAlgorithmPara
meterException
                | CertificateException | KeyStoreException | IOExcep
tion e) {
                throw new RuntimeException("failed to generate a key"
, e);
            }
        }

        private boolean initializeCipherObject() {
            try {
                mKeyStore.load(null);
                SecretKey key = (SecretKey) mKeyStore.getKey(KEY_NAM
E, null);
                SecretKeyFactory factory = SecretKeyFactory.getInsta
nce(KeyProperties.KEY_ALGORITHM_AES, PROVIDER_NAME);
                KeyInfo info = (KeyInfo) factory.getKeySpec(key, Key
Info.class);
                mCipher.init(Cipher.ENCRYPT_MODE, key);
                return true;
            } catch (KeyPermanentlyInvalidatedException e) {
                // *** POINT 6 *** Design your app on the assumption
 that the status of fingerprint registration will change between
 when keys are created and when keys are used
                return false;
            } catch (KeyStoreException | CertificateException
                | UnrecoverableKeyException | IOException
                | NoSuchAlgorithmException | InvalidKeySpecException

                | NoSuchProviderException | InvalidKeyException e) {
                throw new RuntimeException("failed to init Cipher",
e);
            }
        }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="authentication.fingerprint.android.jssec.org.finger
printauthentication" >
    <!-- +++ POINT 1 *** Declare the use of the USE_FINGERPRINT
permission -->
    <uses-permission android:name="android.permission.USE_FINGER
PRINT" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:screenOrientation="portrait" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />
                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

5.7.2 规则书

使用指纹认证的时候，遵循下列规则：

5.7.2.1 创建（注册）密钥时，请使用没有漏洞的加密算法（符合标准）（必需）

与“5.6 使用密码学”中讨论的密码密钥和公密一样，使用指纹认证功能来创建密钥时，必须使用没有漏洞的加密算法 - 即符合某些标准的算法，来防止第三方的窃听。 事实上，安全和没有漏洞的选择不仅适用于加密算法，而且适用于加密模式和填充。

算法选择的更多信息，请参见“5.6.2.2 使用强算法（特别是符合相关标准的算法）（必需）”部分。

5.7.2.2 将加密数据限制为，可通过指纹认证以外的方法恢复（替换）的东西（必需）

当应用使用指纹认证功能，对应用中的数据进行加密时，应用的设计必须允许通过指纹认证以外的方法恢复（替换）数据。 一般来说，使用生物信息会带来各种问题 - 包括保密性，修改难度和错误识别 - 因此，最好避免单纯依靠生物信息进行认证。

例如，假设应用内部的数据使用密钥加密，密钥由指纹认证功能生成，但存储在终端内的指纹数据随后会被用户删除。 然后用于加密数据的密钥不可用，也不可能复制数据。 如果数据不能通过指纹认证功能以外的某种方式恢复，则存在数据无法使用的巨大风险。

此外，指纹信息的删除不是唯一的情况，即使用指纹认证功能创建的密钥可能变得不可用。 在 Nexus5X 中，如果使用指纹认证功能来创建密钥，然后将该密钥注册为额外的指纹信息，则据观察，之前创建的密钥不可用 [30]。此外，不能排除这种可能性，由于指纹传感器的错误识别，通常可以正确使用的密钥变得不可用。

> [30] 信息来自 2016 年 9 月 1 日的版本。 这可能会在未来进行修改。

5.7.2.3 通知用户需要注册指纹才能创建密钥（推荐）

为了使用指纹认证创建密钥，有必要在终端上注册用户的指纹。 设计应用来引导用户进入设置菜单来鼓励指纹注册时，开发人员必须记住，指纹代表重要的个人数据，并且希望向用户解释为什么应用使用指纹信息是必要的或便利的。

通知用户需要注册指纹

```
if (!mFingerprintAuthentication.isFingerprintAuthAvailable()) {
    // **Point** Notify users that fingerprint registration will
 be required to create a key
    new AlertDialog.Builder(this)
        .setTitle(R.string.app_name)
        .setMessage("No fingerprint information has been registe
red.¥n" +
            " Click ¥"Security¥" on the Settings menu to registe
r fingerprints.¥n" +
            " Registering fingerprints allows easy authenticatio
n.")
        .setPositiveButton("OK", null)
        .show();
    return false;
}
```

5.7.3 高级话题

5.7.3.1 Android 应用使用指纹认证功能的先决条件

为了让应用使用指纹认证，必须满足以下两个条件。

- 用户指纹必须在终端内注册。

- （特定于应用的）密钥必须关联注册的指纹。

注册用户指纹

用户指纹信息只能通过设置菜单中的“安全”选项进行注册；一般应用不能执行指纹注册过程。 因此，如果应用尝试使用指纹认证功能时未注册指纹，则应用必须引导用户进入设置菜单并鼓励用户注册指纹。 此时，应用需要向用户提供一些解释，说明为什么使用指纹信息是必要和方便的。

另外，作为指纹注册的必要前提条件，终端必须配置一个替代的屏幕锁定机制。 在指纹已在终端中注册的状态下，如果屏幕锁定被禁用，注册的指纹信息将被删除。

创建和注册密钥

为了关联密钥和终端中注册的指纹，请使用由 `AndroidKeyStore` 供应器提供的 `KeyStore` 实例，来创建并注册新密钥或注册现有密钥。 为了创建关联指纹信息的密钥，请在创建 `KeyGenerator` 时配置参数设置，来启用用户认证请求。

创建并注册关联指纹信息的密钥

```
try {
    // Obtain an instance from the "AndroidKeyStore" Provider
    KeyGenerator keyGenerator = KeyGenerator.getInstance(KeyProp
erties.KEY_ALGORITHM_AES, "AndroidKeyStore");
    keyGenerator.init(
        new KeyGenParameterSpec.Builder(KEY_NAME, KeyProperties.
PURPOSE_ENCRYPT)
            .setBlockModes(KeyProperties.BLOCK_MODE_CBC)
            .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADD
ING_PKCS7)
            .setUserAuthenticationRequired(true) // Enable reque
sts for user (fingerprint) authentication
            .build());
    keyGenerator.generateKey();
} catch (IllegalStateException e) {
    // no fingerprints have been registered in this terminal
    throw new RuntimeException(“No fingerprint registered”, e);
} catch (NoSuchAlgorithmException | InvalidAlgorithmParameterExc
eption
    CertificateException | KeyStoreException | IOException e) {
    // failed to generate a key
    throw new RuntimeException("Failed to generate a key", e);
}
```

为了关联指纹信息和现有密钥，请使用 `KeyStore` 条目，将该密钥注册到已添加设置的东西，来启用用户认证请求。

关联指纹信息和现有密钥

```
SecretKey key = …; // existing key
KeyStore keyStore = KeyStore.getInstance(“AndroidKeyStore”);
keyStore.load(null);
keyStore.setEntry(
    "alias_for_the_key",
    new KeyStore.SecretKeyEntry(key),
    new KeyProtection.Builder(KeyProperties.PURPOSE_ENCRYPT)
        .setUserAuthenticationRequired(true) // Enable requests
for user (fingerprint) authentication
    .build());
```

六、困难问题

在 Android 中，由于 Android 操作系统规范或 Android 操作系统提供的功能，难以确保应用实现的安全性。这些功能被恶意第三方滥用或用户不小心使用，始终存在可能导致信息泄露等安全问题的风险。本章通过指出开发人员可以针对这些功能采取的风险缓解计划，将一些需要引起注意的主题挑选为文章。

6.1 来自剪贴板的信息泄露风险

复制和粘贴是用户经常以不经意的方式使用的功能。例如，不少用户使用这些功能来存储好奇或重要的信息，将邮件或网页中的东西记到记事本中，或者从存储密码的记事本复制并粘贴密码，以便不会提前忘记。这些明显非常随意的行为，但实际上存在用户处理的信息可能被盗的隐藏风险。

这个风险与 Android 系统中的复制粘贴机制有关。用户或应用复制的信息，曾经存储在称为剪贴板的缓冲区中。存储在剪贴板中的信息，在被用户或应用粘贴时，分发给其他应用。所以这个剪贴板功能中存在导致信息泄漏的风险。这是因为剪贴板的实体在系统中是唯一的，并且任何应用都可以使用 `ClipboardManager`，随时获取存储在剪贴板中的信息。这意味着用户复制/剪切的所有信息都会泄露给恶意应用。

因此，考虑到 Android 操作系统的规范，应用开发人员需要采取措施，尽量减少信息泄露的可能性。

6.1.1 示例代码

粗略地说，有两种对策用于减轻来自剪贴板的信息泄露风险

1. 从其他应用复制到你的应用时采取对策。
2. 从你的应用复制到其他应用时采取对策。

首先，让我们讨论上面的对策（1）。假设用户从其他应用（如记事本，Web 浏览器或邮件应用）复制字符串，然后将其粘贴到你的应用的 `EditText` 中。事实证明，在这种情况下，基本没有对策，来防止由于复制和粘贴而导致的敏感信息泄漏。由于 Android 中没有功能来控制第三方应用的复制操作。因此，就对策（1）而言，除了向用户解释复制和粘贴敏感信息的风险外，没有任何方法，只能继续让用户自行减少操作。

接下来的讨论是上面的对策（2），假设用户复制应用中显示的敏感信息。在这种情况下，防止泄漏的有效对策是，禁止来自视图（`TextView`，`EditText` 等）的复制/剪切操作。如果输入/输出敏感信息（如个人信息）的视图中，没有复制/剪切功能，信息泄漏永远不会通过剪贴板在你的应用发生。

有几种禁止复制/剪切的方法。本节介绍简单有效的方法：一种方法是禁用视图的长按，另一种方法是在选择字符串时从菜单中删除复制/剪切条目。

对策的必要性可以根据图 6.1-1 的流程确定。 在图 6.1-1 中，“输入类型固定为密码属性”表示，输入类型在应用运行时必须是以下三种之一。 在这种情况下，由于默认禁止复制/剪切，因此不需要采取对策。

- `InputType.TYPE_CLASS_TEXT | InputType.TYPE_TEXT_VARIATION_PASSWO`
- `InputType.TYPE_CLASS_TEXT | InputType.TYPE_TEXT_VARIATION_WEB_PA`
- `InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_VARIATION_PA`

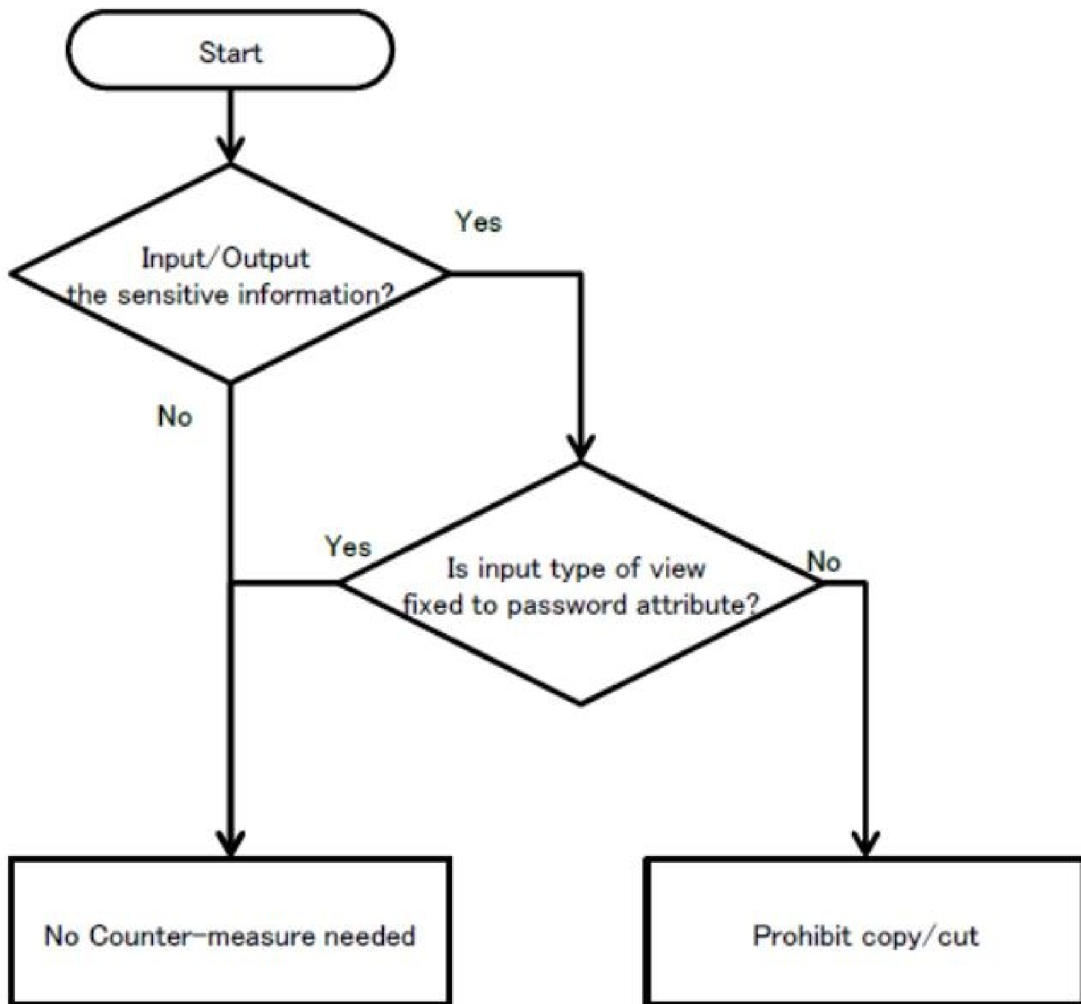


Figure 6.1-1 Decision flow of counter-measure is required or not.

以下小节使用每个示例代码详细介绍了对策。

6.1.1.1 选择字符串时，从菜单中删除复制/剪切条目

在 Android 3.0（API Level 11）之前不能使用 `TextView.setCustomSelectionActionMODECallback()` 方法。 在这种情况下，禁止复制/剪切的最简单方法是禁用视图的长按。 禁用视图的长按可以在 `layout.xml` 文件中规定。

下面展示了示例代码，用于从 `EditText` 中的字符串选择菜单中删除复制/剪切条目。

要点：

1. 从字符串选择菜单中删除 `android.R.id.copy` 。
2. 从字符串选择菜单中删除 `android.R.id.cut` 。

UncopyableActivity.java

```
package org.jssec.android.clipboard.leakage;

import android.app.Activity;
import android.os.Bundle;
import android.support.v4.app.NavUtils;
import android.view.ActionMode;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.EditText;

public class UncopyableActivity extends Activity {

    private EditText copyableEdit;
    private EditText uncopyableEdit;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.uncopyable);
        copyableEdit = (EditText) findViewById(R.id.copyable_edi
t);
        uncopyableEdit = (EditText) findViewById(R.id.uncopyable
_edit);
        // By setCustomSelectionActionMODECallback method,
        // Possible to customize menu of character string select
ion.
        uncopyableEdit.setCustomSelectionActionModeCallback(acti
onModeCallback);
    }

    private ActionMode.Callback actionModeCallback = new ActionM
ode.Callback() {

        public boolean onPrepareActionMode(ActionMode mode, Menu
 menu) {
            return false;
        }

        public void onDestroyActionMode(ActionMode mode) {
        }

        public boolean onCreateActionMode(ActionMode mode, Menu
menu) {
            // *** POINT 1 *** Delete android.R.id.copy from the
 menu of character string selection.
```

```
            MenuItem itemCopy = menu.findItem(android.R.id.copy)
;
            if (itemCopy != null) {
                menu.removeItem(android.R.id.copy);
            }
            // *** POINT 2 *** Delete android.R.id.cut from the
menu of character string selection.
            MenuItem itemCut = menu.findItem(android.R.id.cut);
            if (itemCut != null) {
                menu.removeItem(android.R.id.cut);
            }
            return true;
        }

        public boolean onActionItemClicked(ActionMode mode, Menu
Item item) {
            return false;
        }
    };

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.uncopyable, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                NavUtils.navigateUpFromSameTask(this);
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

6.1.1.2 禁用视图的长按

禁止复制/剪切也可以通过禁用视图的长按来实现。 禁用视图的长按可以在 `layout.xml` 文件中规定。

要点：

1. 在视图中将 `android:longClickable` 设置为 `false` ，来禁止复制/剪切。

unlongclickable.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/unlongclickable_description" />
    <!-- EditText to prohibit copy/cut EditText -->
    <!-- *** POINT 1 *** Set false to android:longClickable in V
iew to prohibit copy/cut. -->
    <EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:longClickable="false"
    android:hint="@string/unlongclickable_hint" />
</LinearLayout>
```

6.1.2 规则书

将敏感信息从你的应用复制到其他应用时，请遵循以下规则：

6.1.2.1 禁用视图中显示的复制/剪切字符串（必需）

如果应用中存在显示敏感信息的视图，并且允许在视图中像 `EditText` 一样复制/剪切信息，信息可能会通过剪贴板泄漏。 因此，必须在显示敏感信息的视图中禁用复制/剪切。 有两种方法禁用复制/剪切。 一种方法是从字符串选择菜单中删除复制/剪切条目，另一种方法是禁用视图的长按。 请参阅“6.1.3.1 应用规则时的注意事项”。

6.1.3 高级话题

6.1.3.1 应用规则时的注意事项

在 `TextView` 中，选择字符串是不可能的，因此通常不需要对策，但在某些情况下，可以复制取决于应用的规范。选择/复制字符串的可能性可以通过使用 `TextView.setTextIsSelectable()` 方法动态决定。将 `TextView` 设置为可以复制时，应调查在 `TextView` 中显示任何敏感信息的可能性，并且如果有任何可能性，则不应将其设置为可复制的。

另外，在“6.1.1 示例代码”的决策流程中描述，根据 `EditText` 的输入类型（`InputType.TYPE_CLASS_TEXT | InputType.TYPE_TEXT_VARIATION_PASSWORD` 等），假设输入类型是密码，通常不需要任何对策，因为复制字符串是默认禁止

的。但是，如“5.1.2.2 提供以明文显示密码的选项（必需）”中所述，如果准备了【以明文显示密码】的选项，则在以明文显示密码的情况下，输入类型将会改变，并且启用复制/剪切。因此应该要求采取同样的对策。

请注意，开发者在应用规则时，还应考虑到应用的可用性。 例如，在用户可以自由输入文本的视图的情况下，如果因输入敏感信息的可能性很小而禁用了复制/剪切，用户可能会感到不便。 当然，该规则应该无条件地，应用于处理非常重要的信息或独立的敏感信息的视图，但在视图之外的情况下，以下问题将帮助开发人员了解如何正确处理视图。

- 准备一些专门用于敏感信息的其他组件
- 当向应用的粘贴是显而易见的时候，用其他方法发送信息
- 提醒用户注意输入/输出信息
- 重新审视视图的必要性

信息泄露风险的根源在于，Android 操作系统中剪贴板和剪贴板管理器的规范不考虑安全风险。 应用开发人员需要在用户完整性，可用性，功能等方面创建更高质量的应用。

6.1.3.2 存储在剪贴板中的操作信息

正如“6.1 来自剪贴板的信息泄漏风险”中所述，应用可以使用 `ClipboardManager` ，操作存储在剪贴板中的信息。另外，不需要为使用 `ClipboardManager` 设置特定的权限，因此应用可以在不被用户识别的情况下，使用 `ClipboardManager` 。

存储在剪贴板中的信息称为 `ClipData` ，可以通过 `ClipboardManager.getPrimaryClip()` 方法获得。如果通过 `ClipboardManager.addPrimaryClipChangedListener()` 方法，将侦听器注册到 `ClipboardManager` ，并实现了 `OnPrimaryClipChangedListener` ，则每次用户执行复制/剪切操作时都会调用监听器。因此可以在不忽略时间的情况下获得 `ClipData` 。在任何应用中执行复制/剪切操作时，都会调用监听器。

下面显示了服务的源代码，无论什么时候在设备中执行复制/剪切，它都会获取 `ClipData` 并通过 `Toast` 显示。你可以意识到，存储在剪贴板中的信息被泄露出来，就是由于下面的简单代码。有必要注意，敏感信息至少不会由以下源代码使用。

ClipboardListeningService.java

```
package org.jssec.android.clipboard;

import android.app.Service;
import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.ClipboardManager.OnPrimaryClipChangedList
ener;
import android.content.Context;
import android.content.Intent;
```

```
import android.os.IBinder;
import android.util.Log;
import android.widget.Toast;

public class ClipboardListeningService extends Service {

    private static final String TAG = "ClipboardListeningService"
;
    private ClipboardManager mClipboardManager;

    @Override
    public IBinder onBind(Intent arg0) {
        return null;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        mClipboardManager = (ClipboardManager) getSystemService(
Context.CLIPBOARD_SERVICE);
        if (mClipboardManager != null) {
            mClipboardManager.addPrimaryClipChangedListener(clip
Listener);
        } else {
            Log.e(TAG, "Failed to get ClipboardService . Service
 is closed.");
            this.stopSelf();
        }
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        if (mClipboardManager != null) {
            mClipboardManager.removePrimaryClipChangedListener(c
lipListener);
        }
    }

    private OnPrimaryClipChangedListener clipListener = new OnPr
imaryClipChangedListener() {

        public void onPrimaryClipChanged() {
            if (mClipboardManager != null && mClipboardManager.h
asPrimaryClip()) {
                ClipData data = mClipboardManager.getPrimaryClip
();

                ClipData.Item item = data.getItemAt(0);
                Toast.makeText(
                    getApplicationContext(),
                    "Character stirng that is copied or cut:¥n"
                        + item.coerceToText(getApplicationContex
t()),
```

```
                    Toast.LENGTH_SHORT)
                    .show();
            }
        }
    };
}
```

接下来，下面显示了`Activity`的示例代码，它使用上面涉及的`ClipboardListeningService`。

ClipboardListeningActivity.java

```
package org.jssec.android.clipboard;

import android.app.Activity;
import android.content.ComponentName;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;

public class ClipboardListeningActivity extends Activity {

    private static final String TAG = "ClipboardListeningActivit
y";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_clipboard_listening);
    }

    public void onClickStartService(View view) {
        if (view.getId() != R.id.start_service_button) {
            Log.w(TAG, "View ID is incorrect.");
        } else {
            ComponentName cn = startService(
                new Intent(ClipboardListeningActivity.this, Clip
boardListeningService.class));
            if (cn == null) {
                Log.e(TAG, "Failed to launch the service.");
            }
        }
    }
    public void onClickStopService(View view) {
        if (view.getId() != R.id.stop_service_button) {
            Log.w(TAG, "View ID is incorrect.");
        } else {
            stopService(new Intent(ClipboardListeningActivity.th
is, ClipboardListeningService.class));
        }
    }
}
```

到目前为止，我们已经介绍了获取存储在剪贴板上的数据的方法。 也可以使用 `ClipboardManager.setPrimaryClip()` 方法在剪贴板上存储新数据。

请注意， `setPrimaryClip()` 方法将覆盖存储在剪贴板中的信息，因此用户的复制/剪切存储的信息可能会丢失。 当使用这些方法提供自定义复制/剪切功能时，必须按需设计/实现，以防止存储在剪贴板中的内容改变为意外内容，通过显示对话框来通知内容将被改变。